

OS/2® Warp Workplace Shell API

Mindy Pollack
Edited by Marc Stock



QA
76
.76
.063
P64
1995

OS/2® Warp

Workplace Shell API

Mindy Pollack

Edited by Marc Stock



John Wiley & Sons, Inc.

New York • Chichester • Brisbane • Toronto • Singapore

Publisher: Katherine Schowalter
Editor: Theresa Hudson
Managing Editor: Robert S. Aronds
Text Design: Tenenbaum Design
Composition: Impressions, A Division of Edwards Brothers, Inc.

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Inc. is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

This text is printed on acid-free paper.

Copyright 1995 by John Wiley & Sons, Inc.

All rights reserved. Published simultaneously in Canada.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional service. If legal advice or other expert assistance is required, the services of a competent professional person should be sought.

Reproduction or translation of any part of this work beyond that permitted by section 107 or 108 of the 1976 United States Copyright Act without the permission of the copyright owner is unlawful. Requests for permission or further information should be addressed to the Permissions Department, John Wiley & Sons, Inc.

ISBN 0-471-03872-5

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

Preface

This reference book covers the programming interface for Workplace Shell in OS/2 versions 2.1 and 3.0. There are two major features that make this book distinctive from other Workplace Shell references—its organization and level of detail.

As with the other books in this series, the information is organized by subject to make this book as usable as possible. This is advantageous for both beginners and experts. A beginner typically knows what functionality he or she wants to implement, but not necessarily the names of the desired methods. Each chapter covers a specific topic; this allows a beginner to quickly access information about all the methods relevant to the task at hand. An advanced programmer typically implements features one category at a time and, therefore, will find this book's topic-oriented organization useful.

This book is written by a Workplace Shell programmer for Workplace Shell programmers, ensuring a high level of detail in the information presented. I have included hints and tips that I have learned through my experiences writing Workplace applications and working on the Workplace Shell development team.

One consequence of the level of detail is that basic information about Presentation Manager (PM) programming, as well as System Object Model (SOM) concepts, is outside the scope of this book. Knowledge of these topics, as well as a complete understanding of C or C++, is a prerequisite for Workplace Shell programming. It is also helpful to have a working knowledge of how to use OS/2 and the Workplace Shell interface. Furthermore, this book does not explain step-by-step how to design a Workplace Shell subclass; instead, the book focuses on providing thorough descriptions of each method. Writing SOM and, especially, Workplace Shell applications can be very difficult at first, and beginners should refer to the WPCar sample included in the *OS/2 Developer's Toolkit* as a guide for designing a subclass.

Notes

- The samples in this book are written using C. C++ programmers can still make use of this code, keeping in mind that the syntax for invoking methods is different. For example:

```
_wpViewObject(Object, NULLHANDLE, OPEN_DEFAULT, 0); /* C syntax */  
Object->wpViewObject(NULLHANDLE, OPEN_DEFAULT, 0); /* C++ syntax */
```

- In C the first parameter of each SOM method is always **somSelf**. Throughout this book the **somSelf** parameter is assumed and is not specifically listed.
- The structure definitions include the offset of each field listed in the right column of the page.

Acknowledgments

This book could not have been completed without the help of many people. Many thanks go to the folks who taught me everything I know and answered my constant barrage of questions as I composed this book—Sheila Harnett, Peter Magid, James Taylor, Dan Kehn, and Chris Andrew. Also to Marc Stock for getting me involved with this series and helping me with the planning stages. Most importantly, thanks to my husband, Mark Engelberg, for supporting me and giving me a reader's perspective.

Table of Contents

Preface	iii
Acknowledgments	v
1 Workplace Shell Overview	1
2 Installing Workplace Classes	6
WinCopyObject	8
WinCreateObject	9
WinCreateShadow	11
WinDeregisterObjectClass	12
WinDestroyObject	13
WinEnumObjectClasses	14
WinMoveObject	16
WinOpenObject	16
WinQueryObject	17
WinRegisterObjectClass	18
WinReplaceObjectClass	19
WinSaveObject	21
WinSetObjectData	22
SysCopyObject	24
SysCreateObject	25
SysCreateShadow	27
SysDeregisterObjectClass	28
SysDestroyObject	29
SysMoveObject	30
SysOpenObject	31
SysQueryClassList	32
SysRegisterObjectClass	33
SysSaveObject	33
SysSetObjectData	34
Structures	36
3 Object Manipulation	37
wpConfirmDelete	40
wpCopiedFromTemplate	43
	vii

	wpCopyObject	44
	wpCreateAnother	47
	wpCreateFromTemplate	48
	wpCreateShadowObject	50
	wpDelete	51
	wpFree	53
	wplsDeleteable	54
	wplsLocked	55
	wplsObjectInitialized	56
	wpLockObject	56
	wpMoveObject	57
	wpObjectReady	58
	wpQueryHandle	60
	wpScanSetupString	61
	wpSetup	63
	wpSetupOnce	64
	wpUnlockObject	65
	wpclsCreateDefaultTemplates	67
	wpclsDecUsage	69
	wpclsIncUsage	70
	wpclsNew	71
	wpclsObjectFromHandle	72
	wpclsQueryAwakeObject	73
	wpclsQueryFolder	74
	wpclsQueryObject	75
	wpclsQueryObjectFromPath	76
4	Instance and Class Data	78
	wpAllocMem	80
	wpFreeMem	82
	wpInitData	83
	wpRestoreData	84
	wpRestoreLong	87
	wpRestoreState	89
	wpRestoreString	90
	wpSaveData	92
	wpSaveDeferred	94
	wpSaveImmediate	95
	wpSaveLong	96
	wpSaveState	97
	wpSaveString	98
	wpUnInitData	100
	wpclsInitData	101
	wpclsUnInitData	102

5	General Object Settings	104
	wpCnrRefreshDetails	106
	wpCnrSetEmphasis	107
	wpModifyStyle	108
	wpQueryConfirmations	111
	wpQueryCoreRecord	112
	wpQueryDetailsData	113
	wpQueryIcon	114
	wpQueryIconData	115
	wpQueryObjectID	117
	wpQueryStyle	118
	wpQueryTitle	118
	wpSetIcon	119
	wpSetIconData	120
	wpSetStyle	121
	wpSetTitle	122
	wpclsQueryDetailsInfo	123
	wpclsQueryIconData	124
	wpclsQuerySetting	126
	wpclsQueryStyle	127
	wpclsQueryTitle	129
	wpclsSetIcon	130
	wpclsSetIconData	130
	wpclsSetSetting	131
	Structures	132
	Details Data Sample Code	138
6	In-Use List	144
	wpAddToObjUseList	145
	wpDeleteFromObjUseList	148
	wpFindUseItem	150
	wpFindViewItem	152
	Structures	153
7	Views	156
	wpClose	158
	wpHide	159
	wpOpen	160
	wpQueryButtonAppearance	162
	wpQueryConcurrentView	164
	wpQueryDefaultView	165
	wpQueryMinWindow	166
	wpRegisterView	167
	wpRestore	168
	wpSetButtonAppearance	169

	wpSetConcurrentView	170
	wpSetDefaultView	171
	wpSetMinWindow	172
	wpSwitchTo	173
	wpViewObject	174
	wpWaitForClose	176
	wpclsQueryButtonAppearance	177
	wpclsQueryDefaultView	178
	wpclsQueryObjectFromFrame	179
8	Settings Notebooks	180
	wpAddSettingsPages	181
	wpInsertSettingsPage	183
	wpclsQuerySettingsPageSize	185
	wpclsSetSettingsPageSize	186
	Structures	187
9	Pop-up Menus	190
	wpDisplayMenu	191
	wpFilterPopupMenu	193
	wpInsertPopupMenuItems	195
	wpMenuItemSelected	198
	wpModifyPopupMenu	199
10	Drag/Drop	200
	wpDraggedOverObject	202
	wpDragOver	204
	wpDrop	206
	wpDroppedOnObject	209
	wpFormatDragItem	209
	Structures	211
11	File System Objects	213
	wpQueryAttr	215
	wpQueryCreation	216
	wpQueryEASize	217
	wpQueryFileSize	218
	wpQueryLastAccess	219
	wpQueryLastWrite	219
	wpQueryRealName	220
	wpQueryType	221
	wpRefresh	222
	wpSetAttr	223
	wpSetType	224
	wpVerifyUpdateAccess	225
	wpclsQueryInstanceType	226
	wpclsQueryInstanceFilter	227

	Structures	228
12	Folder Objects	229
	wpAddFirstChild	231
	wpAddToContent	232
	wpCnrInsertObject	233
	wpCnrRemoveObject	234
	wpContainsFolders	235
	wpDeleteContents	236
	wpDeleteFromContent	237
	wplsCurrentDesktop	238
	wplsDetailsColumnVisible	239
	wplsSortAttribAvailable	240
	wpModifyFldrFlags	241
	wpPopulate	242
	wpQueryContent	243
	wpQueryFldrAttr	245
	wpQueryFldrDetailsClass	247
	wpQueryFldrFlags	248
	wpQueryFldrFont	248
	wpQueryFldrSortClass	250
	wpQueryFolder	251
	wpQueryNextIconPos	251
	wpSetDetailsColumnVisibility	252
	wpSetFldrAttr	253
	wpSetFldrDetailsClass	254
	wpSetFldrFlags	255
	wpSetFldrFont	256
	wpSetFldrSortClass	257
	wpSetSortAttribAvailable	258
	wpcIsQueryActiveDesktop	259
	wpcIsQueryActiveDesktopHWND	260
	wpcIsQueryIconDataN	260
	wpcIsQueryIconN	262
	wpcIsQueryOpenFolders	262
	wpcIsInsertMultipleObjects	264
	wpcIsRemoveObjects	265
	Structures	267
13	Programs and Associations	268
	wpConfirmKeepAssoc	270
	wpQueryAssociatedFileIcon	271
	wpQueryAssociatedProgram	272
	wpQueryAssociationFilter	275
	wpQueryAssociationType	276

	wpQueryProgDetails	277
	wpQueryScreenGroupID	279
	wpSetAssociatedFileIcon	280
	wpSetAssociationFilter	281
	wpSetAssociationType	282
	wpSetProgDetails	283
	Structures	284
14	Disk Objects	286
	wpEjectDisk	287
	wplsDiskSwapped	288
	wpLockDrive	289
	wpQueryDisk	290
	wpQueryDriveLockStatus	291
	wpQueryLogicalDrive	292
	wpQueryRootFolder	293
	wpSetCorrectDiskIcon	293
15	Shadow Objects	295
	wpCreateShadowObjectExt	296
	wpQueryShadowedObject	298
	wpSetLinkToObject	299
	wpSetShadowTitle	300
16	Launch Pad Objects	301
	wpQueryActionButtonButtons	303
	wpQueryActionButtonStyle	304
	wpQueryCloseDrawer	305
	wpQueryDisplaySmallIcons	305
	wpQueryDisplayText	306
	wpQueryDisplayTextInDrawers	307
	wpQueryDisplayVertical	308
	wpQueryDrawerHWND	308
	wpQueryFloatOnTop	309
	wpQueryHideLaunchPadFrameCtrls	310
	wpQueryObjectList	311
	wpRefreshDrawer	312
	wpSetActionButtonStyle	313
	wpSetCloseDrawer	314
	wpSetDisplaySmallIcons	314
	wpSetDisplayText	315
	wpSetDisplayTextInDrawers	316
	wpSetDisplayVertical	317
	wpSetDrawerHWND	317
	wpSetFloatOnTop	318

	wpSetHideLaunchPadFrameCtrls	319
	wpSetObjectListFromHObjects	320
	wpSetObjectListFromObjects	321
	wpSetObjectListFromStrings	322
	Structures	323
17	Palette Objects	325
	wpDragCell	327
	wpEditCell	328
	wpPaintCell	329
	wpRedrawCell	330
	wpQueryPalettInfo	330
	wpSelectCell	331
	wpSetPalettInfo	332
	wpSetupCell	333
	wpcIsQueryEditString	334
	Structures	335
18	Help	337
	wpDisplayHelp	338
	wpMenuItemHelpSelected	339
	wpQueryDefaultHelp	340
	wpSetDefaultHelp	341
	wpcIsQueryDefaultHelp	342
19	Printing	344
	wpDeleteAllJobs	346
	wpHoldPrinter	346
	wpJobAdded	347
	wpJobChanged	348
	wpJobDeleted	348
	wpPrintMetaFile	349
	wpPrintObject	350
	wpPrintPifFile	358
	wpPrintPlainTextFile	359
	wpPrintPrinterSpecificFile	360
	wpPrintUnknownFile	361
	wpReleasePrinter	362
	wpQueryComputerName	363
	wpQueryPrinterName	363
	wpQueryQueueOptions	364
	wpQueryRemoteOptions	365
	wpSetDefaultPrinter	366
	wpSetQueueOptions	367
	wpSetRemoteOptions	368
	Structures	369

20	Miscellaneous Methods	371
	wpFindMinWindow	372
	wpModuleForClass	373
	wpQueryError	374
	wpReplacementIsInEffect	375
	wpSetError	375
	wpclsFindObjectEnd	376
	wpclsFindObjectFirst	377
	wpclsFindObjectNext	382
	wpclsFindObjectOne	383
	wpclsQueryError	384
	wpclsSetError	385
21	System Object Model (SOM)	386
	SOMInitModule	387
	SOMOutCharRoutine	388
	somIdFromString	389
	somLPrintf	390
	somPrintf	391
	<class name>GetData	392
	<class name>MethodDebug	393
	<classname>NewClass	394
	<class name>	395
	somGetClass	395
	somGetClassName	396
	somIsA	397
	somIsInstanceOf	397
	somDescendedFrom	398
	somLocateClassFile	399
Appendix A	Setup String Keynames	401
Appendix B	DOS Settings Keynames	409
Appendix C	Settings Page Methods	415
Appendix D	Workplace Default Object IDs	418
Appendix E	Predefined Views	420
Appendix F	Device Object Settings	421
Index		427

●1

Workplace Shell Overview

What Is a Workplace Shell Application?

To the user, Workplace Shell is the entire user interface of OS/2. This includes folders, objects, notebooks, containers, drag/drop, and so on. From a programming perspective, this view is not entirely accurate. The Presentation Manager (PM) APIs provided by OS/2's Window Manager provides the basic GUI programming environment necessary for writing an application. This includes the 1.x controls such as frames, buttons, scroll bars, and list boxes as well as the newer controls such as notebooks, containers, and drag/drop. Therefore, any PM application can take advantage of these controls without necessarily being a Workplace Shell application. Even the Workplace Shell itself, for example, is just a special PM application that is automatically run when the operating system starts.

The Workplace Shell was implemented using IBM's System Object Model (SOM), a language-independent interface that allows programs to be written using object-oriented techniques. The Workplace Shell

API makes available to the programmer all the public classes implemented in the user interface (folders, programs, data files), the ability to subclass them, and the means to reuse and modify their behavior. Therefore, to a programmer, a Workplace Shell application is one that takes advantage of existing Workplace classes by using SOM to create a subclass. This subclass becomes a fully integrated part of the shell with its own unique behavior. The benefit of creating a Workplace application over a PM application is that a Workplace Shell object can make use of the pop-up menu, settings notebook, and drag/drop behavior already implemented by its parent class.

Workplace Shell subclasses are written using SOM and C or C++ and are then compiled into Dynamic Link Libraries (DLLs). Any user-defined class can be added to the Workplace Shell interface by registering the DLL that contains the class definition source code using `WinRegisterObjectClass` (pg. 18). When a Workplace class is loaded, it automatically becomes part of the `PMSHELL.EXE` process.

Predefined Workplace Classes

Since Workplace Shell programming is all about subclassing existing classes, it is imperative to be aware of all the classes provided and how they behave. Because Workplace classes derive behavior from one another, they are arranged in a hierarchical fashion. All Workplace classes are descendants of `WPObject`, the class that provides the default basic behavior of all objects. (Note that all predefined Workplace classes that ship with OS/2 have the prefix 'WP'.) There are three direct subclasses of `WPObject`; these base classes provide different storage mechanisms for the data of each of its objects (see Figure 1.1).

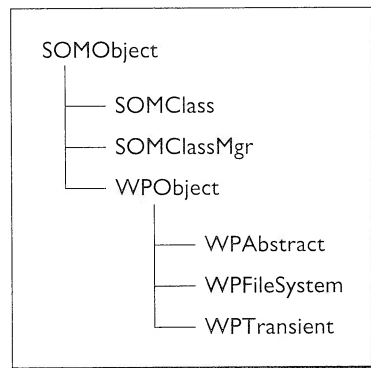


FIGURE 1.1
Workplace base class hierarchy.

Since the base classes define storage mechanisms, they are sometimes referred to as storage classes.

WPAbstract	Object instance data is stored in the user ini file (OS2.INI). They are not stored anywhere else in the file system.
WPFileSystem	Objects represent real files or directories that reside in the directory structure that matches the parent folder hierarchy. Instance data is stored in the Extended Attributes (EAs) of the file.
WPTransient	Objects are not persistent and do not save their instance data. When the machine is rebooted, the objects will not reappear.

Instances of WPObject or the three base classes cannot be created. All other predefined Workplace classes can be instantiated. Defining your own base class is currently not supported in OS/2, therefore, all Workplace classes are descendants of these base classes (see Figure 1.2).

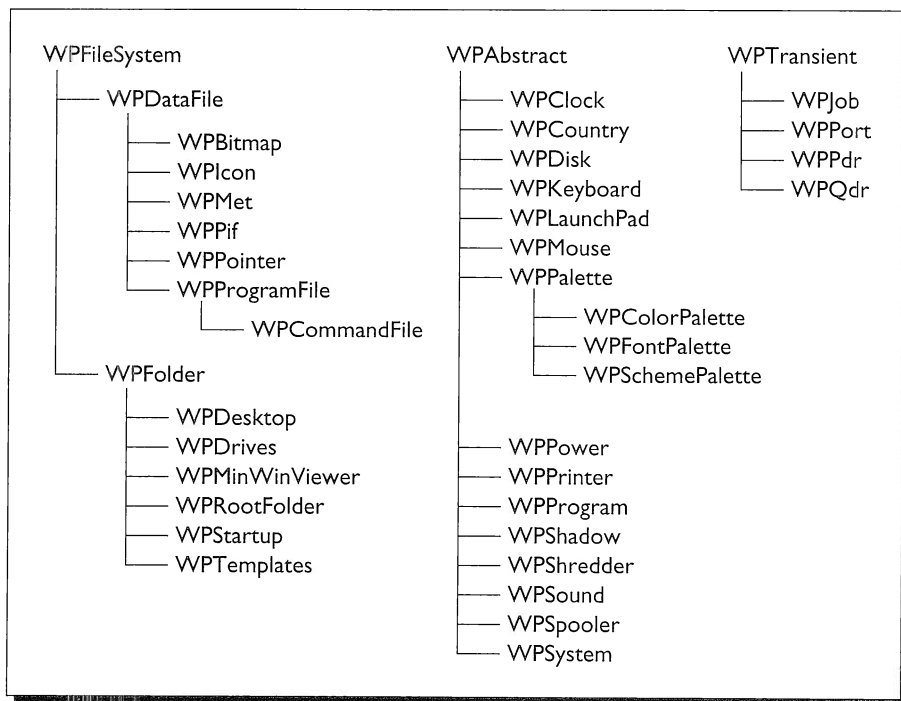


FIGURE 1.2
Workplace class hierarchy.

A lot of thought should be given to choosing the best base class from which to inherit. If instances will be created for temporary use and can be re-created easily, the class should inherit from `WPTransient`. Also consider whether it will be necessary to move instances onto diskettes or across the LAN. `WPFileSystem` is the only base class whose descendants have this capability. `WPAbstract` should be subclassed if instances must persist after the machine has been rebooted but would not necessarily make sense to be in the file system.

All Workplace classes provide new behavior different from their respective parent classes (see Figure 1.2). Some classes do not define public methods; however, instances of such classes can still be created, and methods from their parent classes can be invoked. Furthermore, many of these classes define setup string keywords (see Appendix A) to enable the application to customize their instances.

The device classes `WPMouse`, `WPKeyboard`, and `WPSystem` do not define public methods other than the settings page methods listed in Appendix C. However, they do provide setting strings that can be passed to `wpclsQuerySetting` (pg. 126) and `wpclsSetSetting` (pg. 131).

Replacing a Class

Classes can be added to Workplace or can completely replace existing classes. Replacing a class forces all existing objects of the replaced class to take on the behavior of the user-defined class. (See `WinReplaceObjectClass` pg. 19) The classes `WPMouse`, `WPCountry`, `WPKeyboard`, `WPSound`, `WPClock`, `WPShreder`, `WPPower`, and `WPSystem` are the device classes, each of which is not reentrant and has exactly one instance that cannot be deleted. Consequently, these classes must be replaced in order to modify their behavior. Be aware that if multiple instances of these device classes are created, no error message is given, but unpredictable results will occur and will probably confuse the user.

General Restrictions and Warnings

- You cannot call a class method on an instance pointer or an instance method on a class object pointer. Class methods in the Workplace Shell are named with the prefix 'wpcls' and instance methods have the prefix 'wp'. For example, if you override `wpclsInitData`, the `somSelf` parameter is a class object pointer, and you cannot use it to call an instance method, such as `wpAllocMem` (a common mistake

among first time Workplace Shell programmers). If a class pointer is needed to invoke a class method, call `somGetClass` (pg. 345) on an instance pointer.

- Workplace is designed such that all objects must be created inside a folder. If an object is not designed to be contained in a folder, it can be created in the Nowhere folder, or `<WP_NOWHERE>`, where it will not be visible.
- When calling methods from an application-defined thread, you must first create a message queue for that thread.
- Since all classes run under the Workplace process, it is good practice to keep processing in method overrides short or multithreaded. It is also recommended that classes implement their own exception handling to prevent the entire shell from trapping when an exception occurs.

2

Installing Workplace Classes

Workplace classes cannot be used until they have been registered with the Workplace Shell. Unfortunately, the OS/2 default system does not come with a mechanism for users to register classes. Therefore, the developer of the class DLL must provide this functionality with an installation program.

An installation program for a Workplace class will generally be a stand-alone Presentation Manager executable or a REXX program. Most installation programs will first use Control Program functions to copy the class DLL onto the user's machine, and then call `WinRegisterObjectClass` to register the class with the system. `WinRegisterObjectClass` accepts a fully qualified path of the class DLL location; therefore, the class can either be installed in the `LIBPATH` of the system or in any directory. Once the class has been successfully reg-

istered, the installation program can also use `WinCreateObject` to create instances of the class or other classes on the desktop.

A deinstall program should also be provided to enable the user to deregister the class. See `WinDeregisterObjectClass` for more information.

Manipulating Objects from a Separate Process

All Workplace Shell classes execute in the same `PMSHELL.EXE` process and can therefore manipulate other Workplace objects using the SOM methods described in other chapters in this book. There are a small number of functions that work across processes and can be called from any Presentation Manager executable. These functions, listed in this chapter, allow programs to register and deregister classes; create, destroy, copy, and move objects; and set object data. Most of the functions correspond to Workplace SOM methods and have REXX equivalents provided in the `REXXUTIL.DLL` that ships with OS/2.

Object Handles and IDs

When a Workplace object is created using `WinCreateObject`, it is assigned a unique number called an object handle. These handles are persistent across boots of the machine and can always be used to access the objects they refer to. The object handle is used as a parameter or return code for many of the Workplace APIs to enable applications to uniquely identify the objects.

Objects can also be assigned object IDs as identifiers. Object IDs are strings beginning with '<' and ending with '>'. For example, the object ID for the desktop is '<WP_DESKTOP>'. Applications assign object IDs by using the `OBJECTID` keyname in the setup string parameter. Since object IDs are generated by the application, they are not guaranteed to always be unique. If you use an object ID that is a duplicate, you risk replacing an existing object or mistaking a different object for your own. Applications should include their application names in the object IDs to help ensure uniqueness.

Restrictions and Warnings

- Currently there is no interface for querying object data from a separate executable.

- Object IDs should be used sparingly because they are application-generated strings that must be unique in the system. Whenever possible, save the object handles instead.
- The REXX Workplace APIs accept only object IDs as parameters. Therefore, REXX programs must use object IDs to uniquely identify objects.
- There is no REXX API for replacing an object class.

PM Functions

WinCopyObject (Warp) copies a Workplace object (pg. 8).

WinCreateObject creates a Workplace object (pg. 9).

WinCreateShadow (Warp) creates a shadow of a Workplace object (pg. 11).

WinDeregisterObjectClass deregisters a Workplace class (pg. 12).

WinDestroyObject deletes a Workplace object (pg. 13).

WinEnumObjectClasses enumerates all registered Workplace classes (pg. 14).

WinMoveObject (Warp) moves a Workplace object to another folder (pg. 16).

WinOpenObject (Warp) opens a view of a Workplace object (pg. 16).

WinQueryObject returns the handle of an object (pg. 17).

WinRegisterObjectClass registers a Workplace object class (pg. 18).

WinReplaceObjectClass replaces an existing Workplace class (pg. 19).

WinSaveObject (Warp) notifies an object to save its data (pg. 21).

WinSetObjectData sets instance data on an object (pg. 22).

● **WinCopyObject** (Warp only)

Copies a Workplace object.

SYNTAX

HOBJECT WinCopyObject (**HOBJECT** *hObjectOfObject*, **HOBJECT** *hObjectOfDest*, **ULONG** *ulReserved*)

PARAMETERS

hObjectOfObject - input

The object handle of the object to be copied. Object handles can be obtained by calling WinQueryObject or by saving the handle when it is returned by WinCreateObject.

hObjectOfDest - input

The object handle of the folder in which to place the new copy.

ulReserved - reserved

Set this parameter to 0.

RETURNS

(nonzero value) - The object handle of the new copy.

NULLHANDLE - The copy operation failed.

OTHER INFO

Include file: pmwp.h Define: INCL_WINWORKPLACE

SEE ALSO

WinCreateObject -9, WinCreateShadow -11, WinMoveObject -16,
WinQueryObject -17, wpCopyObject -44

NOTES

If an object exists in the destination folder with the same title as the new copy, the copy operation will fail.

WinCreateObject

Creates a Workplace object of the specified class.

SYNTAX

HOBJECT WinCreateObject (**PSZ** *pszClassName*, **PSZ** *pszTitle*, **PSZ** *pszSetupString*, **PSZ** *pszLocation*, **ULONG** *ulFlag*)

PARAMETERS

pszClassName - input

The name of the class for the new object. This must be a registered Workplace Shell class. See the class hierarchy (pg. 3) for the list of pre-defined Workplace Shell classes. This parameter is case-sensitive.

pszTitle - input

The title for the new object.

pszSetupString - input

The initial settings for the new object. A setup string is a list of key-name value pairs separated by semicolons which is parsed by the object for initialization. Specify NULL to use the object's defaults. See Appendix A for the list of possible keyname value pairs for the predefined Workplace Shell classes. For more information on setup strings, see wpSetup (pg. 63).

pszLocation - input

The folder location for the new object. This can either be the object ID (if it exists and is known) or the fully qualified path of the directory represented by a folder. For example, to create an object on the Desktop, *pszLocation* would either be "<WP_DESKTOP>" or "C:\DESKTOP" (assuming the Desktop is on the C drive and is called 'Desktop'). See Appendix D for a list of predefined folder object IDs.

ulFlag - input

This flag specifies what action should be taken if a duplicate existing object is found. If an object ID is specified in *pszSetupString* and an existing object with that ID is found or an existing object with the same title in the same folder is found, that object is considered a duplicate.

Constant		Description
CO_FAILIFEXISTS	0x00	If a duplicate is found, the new object is not created and the call will fail.
CO_REPLACEIFEXISTS	0x01	If a duplicate is found, the new object is created and replaces the duplicate.
CO_UPDATEIFEXISTS	0x02	If a duplicate is found, the new object is not created and, if specified, <i>pszSetupString</i> is passed to the duplicate as if WinSetObjectData were called.

RETURNS

(nonzero value)—The handle to the newly created object.

NULLHANDLE—object creation failed.

OTHER INFO

Include file: pmwp.h Define: INCL_WINWORKPLACE

SEE ALSO

WinDestroyObject -13, WinRegisterObjectClass -18,
WinSetObjectData -22, wpSetup -63, wpclsNew -71

NOTES

The class name specified in *pszClassName* must be a registered Workplace Shell class. You must use either a predefined Workplace Shell class or a class that has been registered with WinRegisterObjectClass.

When this function is called, Workplace Shell will call wpclsNew on the specified class object to create the new instance.

SAMPLE CODE

```
#define INCL_WINWORKPLACE
#include <os2.h>

HOBJECT hObject;

/* create a folder object on the desktop named 'My Folder' that is
   not deleteable
   * and assign it the object ID, <MY_FOLDER>.
   */
hObject = WinCreateObject ("WPFolder",      //class name
                          "My Folder",      // title of new object
                          "<WP_DESKTOP>",    // place object on the
                                          desktop
                          "NODELETE=YES;OBJECTID=<MY_FOLDER>",
                          CO_FAILIFEXISTS);
```

● **WinCreateShadow (Warp only)**

Creates a shadow of a Workplace object.

SYNTAX

HOBJECT WinCreateShadow (**HOBJECT** *hObjectOfObject*, **HOBJECT** *hObjectOfDest*, **ULONG** *ulReserved*)

PARAMETERS

hObjectOfObject - input

The object handle of the object to be shadowed. Object handles can be obtained by calling WinQueryObject or by saving the handle when it is returned by WinCreateObject.

hObjectOfDest - input

The object handle of the folder in which to place the new shadow.

ulReserved - reserved

Set this parameter to 0.

RETURNS

(nonzero value)—The object handle of the new shadow.

NULLHANDLE—The operation failed.

OTHER INFO

Include file: pmwp.h

Define: INCL_WINWORKPLACE

SEE ALSO

WinCopyObject -8, WinCreateObject -9, WinMoveObject -16,
WinQueryObject -17, wpCreateShadowObject -50

NOTES

If an object exists in the destination folder with the same title as the new shadow, the operation will fail.

● WinDeregisterObjectClass

Deregisters an application-defined Workplace class.

SYNTAX

BOOL WinDeregisterObjectClass (**PSZ** *pszClassName*)

PARAMETERS

pszClassName - input

The name of the class to deregister. Predefined Workplace Shell classes cannot be deregistered. This parameter is case-sensitive.

RETURNS

TRUE—The class deregistered successfully.

FALSE—An error occurred.

OTHER INFO

Include file: pmwp.h Define: INCL_WINWORKPLACE

SEE ALSO

WinDestroyObject -13, WinEnumObjectClasses -14,
WinRegisterObjectClass -18

NOTES

This function should be used to deregister Workplace Shell classes that have been registered using WinRegisterObjectClass. Classes should be deregistered by deinstall programs for applications. When the class is deregistered, the DLL is unloaded. This function will fail if any objects of the class are awake. See Chapter 3 for more information on what it means for an object to be awake.

RESTRICTIONS

It is the application's responsibility to first remove any objects of the class to be deregistered.

WinDestroyObject

Deletes a Workplace object.

SYNTAX

BOOL WinDestroyObject (**HOBJECT** *hObject*)

PARAMETERS

hObject - input

The persistent handle of any object. Even if the object has the style `OBJSTYLE_NODELETE`, WinDestroyObject will still delete it (object styles only prevent the *user* from manipulating objects). Object handles can be obtained by calling WinQueryObject or by saving the handle when it is returned by WinCreateObject.

RETURNS

TRUE—The object was successfully deleted.
FALSE—An error occurred.

OTHER INFO

Include file: pmwp.h Define: INCL_WINWORKPLACE

SEE ALSO

WinCreateObject -9, WinQueryObject -17, wpFree -53,
wpModifyStyle -108

NOTES

When this function is called, Workplace Shell will call wpFree on the object to delete it. If wpFree fails, the call to WinDestroyObject will return FALSE.

Folders will automatically delete their contents when they are destroyed. (See wpFree for details.)

SAMPLE CODE

```
#define INCL_WINWORKPLACE
#include <os2.h>
HOBJECT hObject;
BOOL bSuccess;
hObject = WinQueryObject("<WP_GAMES>"); /* get the handle of the
                                         games folder */

if (hObject)
    bSuccess = WinDestroyObject(hObject); /* now delete it */
```

● **WinEnumObjectClasses**

Enumerates all registered Workplace classes.

SYNTAX

BOOL WinEnumObjectClasses (**POBJCLASS** *pObjClass*, **PULONG** *pSize*)

PARAMETERS

pObjClass - input/output

A pointer to a block of data that will be filled with a linked list of OBJCLASS structures (pg. 36). If a NULL value is passed, the required size for the buffer is returned in *pSize*.

pSize - input/output

If *pObjClass* is nonzero, *pSize* should be set to point to the size of the *pObjClass* buffer. If *pObjClass* is NULL, the required size for the buffer is returned.

RETURNS

TRUE—The call was successful.

FALSE—An error occurred.

OTHER INFO

Include file: pmwp.h Define: INCL_WINWORKPLACE

SEE ALSO

WinDeregisterObjectClass -12, WinRegisterObjectClass -18

SAMPLE CODE

```
#define INCL_WINWORKPLACE
#include <os2.h>
POBJCLASS pObjClass, pCurrentClass;
ULONG      ulSize;
BOOL       bSuccess;
/* first call WinEnumObjectClasses to get the required size for
 * the buffer.
 */
bSuccess = WinEnumObjectClasses(NULL, &ulSize);
if (bSuccess && ulSize)
{
    pObjClass = (POBJCLASS) malloc(ulSize);
    if (pObjClass)
    {
        bSuccess = WinEnumObjectClasses(pObjClass, &ulSize);
        /* access the buffer and free it when finished */
        for (pCurrentClass = pObjClass; pCurrentClass;
            pCurrentClass = pCurrentClass->pNext)
        {
            /* insert code to access class structure pCurrentClass is
             pointing to */
        }
        /* free the pointer */
        free(pObjClass)
    }
}
```

● **WinMoveObject** (Warp only)

Moves a Workplace object.

SYNTAX

HOBJECT WinMoveObject (**HOB**JECT *hObjectOfObject*, **HOB**JECT *hObjectOfDest*, **U**LONG *ulReserved*)

PARAMETERS

hObjectOfObject - input

The object handle of the object to be moved. Object handles can be obtained by calling WinQueryObject or by saving the handle when it is returned by WinCreateObject.

hObjectOfDest - input

The object handle of the new folder location for the specified object.

ulReserved - reserved

Set this parameter to 0.

RETURNS

(nonzero value)—The object handle of the object moved (same as *hObjectOfObject*).

NULLHANDLE—The move operation failed.

OTHER INFO

Include file: pmwp.h

Define: INCL_WINWORKPLACE

SEE ALSO

WinCopyObject -8, WinCreateObject -9, WinCreateShadow -11,
WinQueryObject -17, wpMoveObject -16

NOTES

If an object exists in the destination folder with the same title, the move operation will fail.

● **WinOpenObject** (Warp only)

Opens a view of a Workplace object.

SYNTAX

BOOL WinOpenObject (**HOBJECT** *hObject*, **ULONG** *ulView*, **BOOL** *Flag*)

PARAMETERS

hObject - input

The object handle of the object to open. Object handles can be obtained by calling WinQueryObject or by saving the handle when it is returned by WinCreateObject.

ulView - input

The value representing the view to open for this object. See Appendix E for a list of predefined Workplace views. The view constants are defined in wpobject.h.

Flag - input

When set to TRUE, the view will be opened following the window open behavior of the object. If the object does not support concurrent views and the view specified is already opened, the existing view will be surfaced. When set to FALSE, a new view will always be opened.

RETURNS

TRUE—The view was opened successfully.

FALSE—The open operation failed.

OTHER INFO

Include file: pmwp.h and wpobject.h Define: INCL_WINWORKPLACE

SEE ALSO

WinQueryObject -17, WinSetObjectData -22, wpOpen -160, wpViewObject -174

● WinQueryObject

Returns the persistent handle of an object given an object ID.

SYNTAX

HOBJECT WinQueryObject (**PSZ** *pszObject*)

PARAMETERS

pszObject - input

The object ID of an existing Workplace Shell object or the fully qualified path of a WPFileSystem object. For example, to obtain the object handle of the Desktop, `pszObject` would be either “<WP_DESKTOP>” or “C:\DESKTOP” (assuming the Desktop is on the C drive and is called ‘Desktop’). See Appendix D for a list of the object IDs of Workplace predefined objects.

RETURNS

(non zero value)—The class replacement was successful.

NULLHANDLE—The object was not found or an error occurred.

OTHER INFO

Include file: `pmwp.h` Define: `INCL_WINWORKPLACE`

SEE ALSO

`WinCopyObject` -8, `WinCreateObject` -9, `WinCreateShadow` -11,
`WinDestroyObject` -13, `WinMoveObject` -16, `WinOpenObject` -16,
`WinSetObjectData` -22, `WinSaveObject` -21

NOTES

Many functions described in this chapter accept only object handles as parameters. Therefore, `WinQueryObject` can be used to determine object handles when only the object ID or fully qualified path is known.

● WinRegisterObjectClass

Registers an application-defined Workplace class.

SYNTAX

BOOL WinRegisterObjectClass (**PSZ** *pszClassName*, **PSZ** *pszModuleName*)

PARAMETERS

pszClassName - input

The name of the class to register. This parameter is case-sensitive, and the name of the class must be unique in the system.

pszModuleName - input

The name of the DLL that contains the class definition. This can be a fully qualified path or just the name of a DLL in the LIBPATH.

RETURNS

TRUE—The class registered successfully.

FALSE—An error occurred.

OTHER INFO

Include file: `pmwp.h` Define: `INCL_WINWORKPLACE`

SEE ALSO

`WinCreateObject` -9, `WinDeregisterObjectClass` -12,
`WinEnumObjectClasses` -14, `wpPopulate` -242

NOTES

Workplace classes only need to be registered once. The class will be added to the system class table which is stored in the system ini file. When registering a class contained in a DLL not in the system LIBPATH and the full path of the DLL is not specified, `WinRegisterObjectClass` will return 0 but the class will still be added to the class list. This is convenient for applications that place DLLs in a new directory and modify the LIBPATH statement in the CONFIG.SYS to contain this new directory, because this change will not take effect until the next reboot. However, in this situation, instances of this class cannot be created until after a reboot. If you are overriding `wpclsQueryInstanceType` or `wpclsQueryInstanceFilter`, the class must be in the LIBPATH, or a full path must be specified when it is registered.

When `WinRegisterObjectClass` is called and the class DLL is successfully loaded, Workplace calls `wpPopulate` on the Templates folder; this causes a template of the newly registered class to be created (the `wpclsCreateDefaultTemplates` method is called).

● **WinReplaceObjectClass**

Replaces an existing Workplace class with a user-defined class.

SYNTAX

BOOL WinReplaceObjectClass (**PSZ** *pszOldClassName*, **PSZ** *pszNewClassName*, **BOOL** *bReplace*)

PARAMETERS

pszOldClassName - input

The name of the class to replace. The class must be a registered Workplace class. This parameter is case-sensitive.

pszNewClassName - input

The name of the application-defined class that will replace the behavior of *pszOldClassName*. *pszNewClassName* must be a registered Workplace class and must be a **direct** descendant of *pszOldClassName*. This parameter is case-sensitive.

bReplace - input

Set to TRUE to replace *pszOldClassName* with *pszNewClassName*. Set to FALSE to stop replacing *pszOldClassName* with *pszNewClassName* if such a replacement exists.

RETURNS

TRUE—The class replacement was successful.

FALSE—An error occurred.

OTHER INFO

Include file: `pmwp.h` Define: `INCL_WINWORKPLACE`

SEE ALSO

`WinDeregisterObjectClass` -12, `WinEnumObjectClasses` -14,
`WinRegisterObjectClass` -18

NOTES

A class is replaced to change the behavior of all the existing and future objects of that class. It is used to make global changes on objects of a specific class. When you replace the parent class with your class, all instances of the replaced class become instances of your class. Furthermore, all subclasses of the parent class will inherit from your class.

The word replace is misleading because it implies the parent class is removed from the parent chain and replaced with yours. This is not the case. The parent class remains and the new class is inserted after it in the class hierarchy. Furthermore, when an application creates an instance of the replaced class, the new class is used instead. For example, if Classes B, C, and D are subclasses of Class A, and B replaces A, then each time an instance of A is created, it will instead become an instance of Class B. Also, any instances of A's subclasses will inherit from B. See Figures 2.1a and 2.1b.

If a class is replaced multiple times, unique classes specified in *pszNewClassName* for each call to `WinReplaceObjectClass` will be added to the class replacement hierarchy below the most recent replacement.

Class replacement is useful for changing the behavior of classes that can have only one active instance at a time, such as WPDesktop and the device classes.

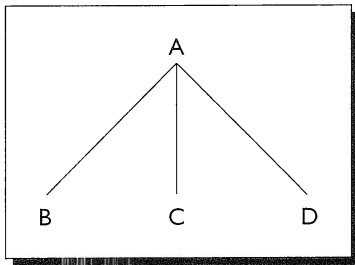


FIGURE 2.1a
Before B replaces A

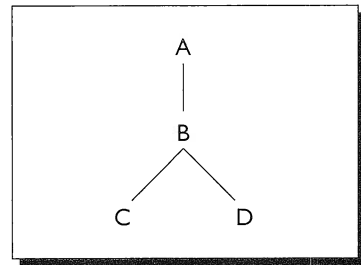


FIGURE 2.1b
After B replaces A

RESTRICTIONS/WARNINGS

- The class specified in *pszOldClassName* must be the direct parent class of *pszNewClassName* as specified in the class definition file (.CSC or .IDL).
- The system must be rebooted before the class replacement can take effect.
- If you replace a class, you run the risk of changing the behavior of existing classes created by other applications. Therefore, class replacement should be carefully thought out and only used when there are no alternatives.

● WinSaveObject (Warp only)

Notifies an object to save its data.

SYNTAX

BOOL WinSaveObject (**HOBJECT** *hObject*, **BOOL** *fAsync*)

PARAMETERS

hObject - input

The object handle of the object to save. Object handles can be obtained by calling WinQueryObject or by saving the handle when it is returned by WinCreateObject.

fAsync - input

If set to TRUE, the object will be saved asynchronously and the function will return immediately. If FALSE, the function will not return until after the object has been saved.

RETURNS

TRUE—The object was saved successfully.

NULLHANDLE—An error occurred.

OTHER INFO

Include file: pmwp.h

Define: INCL_WINWORKPLACE

SEE ALSO

WinQueryObject -17, WinSetObjectData -22, wpSaveDeferred -94, wpSaveImmediate -95

NOTES

When *fAsync* is set to TRUE, the system will call wpSaveDeferred to save the object. When set to FALSE, the system calls wpSaveImmediate.

Normally, when WinSetObjectData is called, the object will automatically save its data asynchronously without calling WinSaveObject. If, however, immediate saving is desired, call WinSaveObject, passing FALSE for *fAsync*.

● WinSetObjectData

Passes setup information to an object.

SYNTAX

BOOL WinSetObjectData (**HOBJECT** *hObject*, **PSZ** *pszSetupString*)

PARAMETERS

hObject - input

The persistent handle of an existing object. Object handles can be obtained by calling WinQueryObject or by saving the handle when it is returned by WinCreateObject.

pszSetupString - input

The desired settings for the specified object. This parameter is in the same format as the *pszSetupString* parameter for WinCreateObject. A setup string is a list of keyname value pairs separated by semicolons

which is parsed by the object to set instance data. See Appendix A for the list of possible keyname value pairs for the predefined Workplace Shell classes. For more information on setup strings, see `wpSetup` (pg. 63).

RETURNS

TRUE—The setup string was successfully passed to the object.

FALSE—The object was not found or an error occurred.

OTHER INFO

Include file: `pmwp.h` Define: `INCL_WINWORKPLACE`

SEE ALSO

`WinCreateObject` -9, `WinQueryObject` -17, `wpSaveDeferred` -94,
`wpSetup` -63

NOTES

When `WinSetObjectData` is called, Workplace Shell passes *pszSetupString* to the object using the `wpSetup` method and then calls `wpSaveDeferred` to asynchronously save the object's data.

`WinSetObjectData` can be used by programs to communicate with objects in the Workplace process without using DSOM. The setup string can be used to communicate with objects that override the `wpSetup` method to manipulate this application-defined data.

RESTRICTIONS

Currently there is no function for querying the settings of an object.

REXX Functions

SysCopyObject (Warp) copies a Workplace object (pg. 24).

SysCreateObject creates a Workplace object (pg. 25).

SysCreateShadow (Warp) creates a shadow of a Workplace object (pg. 27).

SysDeregisterObjectClass deregisters a Workplace class (pg. 28).

SysDestroyObject deletes a Workplace object (pg. 29).

SysMoveObject (Warp) moves a Workplace object to another folder (pg. 30).

SysOpenObject (Warp) opens a view of a Workplace object (pg. 31).

SysQueryClassList lists all registered Workplace classes (pg. 32).

SysRegisterObjectClass registers a Workplace class (pg. 33).

SysSaveObject (Warp) notifies an object to save its data (pg. 33).

SysSetObjectData sets instance data on an object (pg. 34).

● **SysCopyObject REXX (Warp only)**

Copies a Workplace object.

EQUIVALENT C FUNCTION

WinCopyObject

SYNTAX

result = SysCopyObject (*Object*, *Destination*)

PARAMETERS

Object - input

The object to copy. This can be either the object ID (if it exists and is known) or, if the object is a descendant of WPFileSystem, the fully qualified path of the file representing the object.

Destination - input

The folder location for the new copy. This can be either the object ID (if it exists and is known) or the fully qualified path of the directory represented by the folder. For example, to create an object on the Desktop, *Location* would be either '<WP_DESKTOP>' or 'C:\DESKTOP' (assuming the Desktop is on the C drive and is called 'Desktop'). See Appendix D for a list of predefined folder object IDs.

RETURNS

1—The new copy was successfully created.

0—An error occurred.

OTHER INFO

The REXX Workplace functions are contained in REXXUTIL.DLL which must be in the LIBPATH.

The function must be loaded before it can be called. (See sample for SysCreateObject, pg. 25.)

SEE ALSO

SysCreateObject -25, WinCopyObject -8

NOTES

See WinCopyObject (pg. 8) for more information.

● SysCreateObject REXX

Creates a Workplace object of the specified class.

EQUIVALENT C FUNCTION

WinCreateObject

SYNTAX

result = SysCreateObject (*ClassName*, *Title*, *Location* [, *SetupString*, *Flag*])

PARAMETERS

ClassName - input

The name of the class for the new object. This must be a registered Workplace Shell class. See the class hierarchy (pg. 2) for the list of predefined Workplace Shell classes. This parameter is case-sensitive.

Title - input

The title for the new object.

Location - input

The folder location for the new object. This can either be the object ID (if it exists and is known) or the fully qualified path of the directory represented by the folder. For example, to create an object on the Desktop, *Location* would be either '<WP_DESKTOP>' or 'C:\DESKTOP' (assuming the Desktop is on the C drive and is called 'Desktop'). See Appendix D for a list of predefined folder object IDs.

SetupString - input (optional)

The initial settings for the new object. A setup string is a list of key-name value pairs separated by semicolons which is parsed by the object for initialization. Omit to use the object's defaults. See Appendix A for the list of possible keyname value pairs for the predefined Workplace classes. For more information on setup strings, see wpSetup (pg. 63).

Flag - input (optional)

This flag specifies what action should take place if a duplicate existing object is found. If an object ID is specified in *SetupString* and an existing object with that ID is found or an existing object with the same title in the same folder is found, that object is considered a duplicate.

Value	Action
'Fail'	If a duplicate is found, the new object is not created and the call will fail.
'Replace'	If a duplicate is found, the new object is created and replaces the duplicate.
'Update'	If a duplicate is found, the new object is not created and, if specified, <i>SetupString</i> is passed to the duplicate as if <i>SysSetObjectData</i> were called.

Only the first letter of the flag is required. The rest are ignored.

RETURNS

1—The object was successfully created.

0—An error occurred.

OTHER INFO

The REXX Workplace functions are contained in REXXUTIL.DLL which must be in the LIBPATH.

The function must be loaded before it can be called. (See sample code.)

SEE ALSO

SysDestroyObject -29, *SysRegisterObjectClass* -33, *SysSetObjectData* -34, *WinCreateObject* -9

NOTES

SysCreateObject does not return an object handle. The REXX Workplace functions use object IDs only to identify objects.

See *WinCreateObject* (pg. 9) for more information.

SAMPLE CODE

```
/* Load the REXX utility function */
call RxFuncAdd 'SysCreateObject', 'RexxUtil', 'SysCreateObject'

/* create a folder object on the Desktop named 'My Folder' that is
   not deleteable and assign it the object ID, <MY_FOLDER>. Specify
   'Fail' as the last parameter so a duplicate will not be created.
*/
result = SysCreateObject ('WPFolder',,          /* class name */
                        'My Folder',,          /* title */
                        '<WP_DESKTOP>',,
```



```
                                'NODELETE=YES;OBJECTID=<MY_FOLDER>' , ,  
                                'Fail')  
  
if result = 1 then  
    say 'My Folder created successfully'  
else  
    say 'My Folder not created successfully'
```

● **SysCreateShadow** **REXX (Warp only)**

Creates a shadow of a Workplace object.

EQUIVALENT C FUNCTION

WinCreateShadow

SYNTAX

result = SysCreateShadow (*Object*, *Destination*)

PARAMETERS

Object - input

The object to be shadowed. This can be either the object ID (if it exists and is known) or, if the object is a descendant of WPFileSystem, the fully qualified path of the file representing the object.

Destination - input

The folder location for the new shadow. This can be either the object ID (if it exists and is known) or the fully qualified path of the directory represented by the folder. For example, to create an object on the Desktop, *Location* would either be '<WP_DESKTOP>' or 'C:\DESKTOP' (assuming the Desktop is on the C drive and is called 'Desktop'). See Appendix D for a list of predefined folder object IDs.

RETURNS

1—The new shadow was successfully created.
0—An error occurred.

OTHER INFO

The REXX Workplace functions are contained in REXXUTIL.DLL which must be in the LIBPATH.

The function must be loaded before it can be called. (See sample for SysCreateObject, pg. 25.)

SEE ALSO

SysCreateObject -25, WinCreateShadow -11

NOTES

See WinCreateShadow (pg. 11) for more information.

● **SysDeregisterObjectClass** **REXX**

Deregisters a Workplace class.

EQUIVALENT C FUNCTION

WinDeregisterObjectClass

SYNTAX

result = SysDeregisterObjectClass (*ClassName*)

PARAMETERS

ClassName - input

The name of the class to deregister. Predefined Workplace classes cannot be deregistered. This parameter is case-sensitive.

RETURNS

1—The class deregistered successfully.

0—An error occurred.

OTHER INFO

The REXX Workplace functions are contained in REXXUTIL.DLL which must be in the LIBPATH.

The function must be loaded before it can be called. (See sample for SysCreateObject, pg. 25.)

SEE ALSO

WinDeregisterObjectClass -12, SysDestroyObject -29,
SysRegisterObjectClass -33, SysQueryClassList -32

NOTES

See WinDeregisterObjectClass (pg. 12) for more information.

● **SysDestroyObject** **REXX**

Deletes a Workplace object.

EQUIVALENT C FUNCTION

WinDestroyObject

SYNTAX

result = SysDestroyObject (*Object*)

PARAMETERS

Object - input

The Object ID or, if the object is a descendant of WPFileSystem, the fully qualified path of any object. If the object is OBJSTYLE_NODELETE, SysDestroyObject will still try to delete it. See Appendix D for a list of object IDs of the predefined Workplace Shell objects.

RETURNS

1—The object was successfully deleted.

0—An error occurred.

OTHER INFO

The REXX Workplace functions are contained in REXXUTIL.DLL which must be in the LIBPATH. The function must be loaded before it can be called. (See sample code for SysCreateObject, pg. 25.)

SEE ALSO

SysCreateObject -25, WinDestroyObject -13

● **SysMoveObject REXX (Warp only)**

Moves a Workplace object to another folder.

EQUIVALENT C FUNCTION

WinMoveObject

SYNTAX

result = SysMoveObject (*Object*, *Destination*)

PARAMETERS

Object - input

The object to move. This can be either the object ID (if it exists and is known) or, if the object is a descendant of WPFileSystem, the fully qualified path of the file representing the object.

Destination - input

The new folder location for the specified object. This can be either the object ID (if it exists and is known) or the fully qualified path of the directory represented by the folder. For example, to create an object on the Desktop, *Location* would be either '<WP_DESKTOP>' or 'C:\DESKTOP' (assuming the Desktop is on the C drive and is called 'Desktop'). See Appendix D for a list of predefined folder object IDs.

RETURNS

1—The object was moved successfully.

0—An error occurred.

OTHER INFO

The REXX Workplace functions are contained in REXXUTIL.DLL which must be in the LIBPATH. The function must be loaded before it can be called. (See sample for SysCreateObject, pg. 25.)

SEE ALSO

SysCreateObject -25, WinMoveObject -16

NOTES

See WinMoveObject (pg. 16) for more information.

● **SysOpenObject REXX (Warp only)**

Opens a view of a Workplace object.

EQUIVALENT C FUNCTION

WinOpenObject

SYNTAX

result = SysOpenObject (*Object*, *View*, *Flag*)

PARAMETERS

Object - input

The object to open. This can be either the object ID (if it exists and is known) or, if the object is a descendant of WPFileSystem, the fully qualified path of the file representing the object.

View - input

The value representing the view to open for this object. For example, to open the settings view, *View* should be 2, which corresponds to OPEN_SETTINGS. See Appendix E for a list of predefined Workplace views.

Flag - input

When set to 'True', the view will be opened following the window open behavior of the object. If the object does not support concurrent views and the view specified is already opened, the existing view will be surfaced. When set to 'False', a new view will always be opened.

RETURNS

1—The view was opened successfully.

0—An error occurred.

OTHER INFO

The REXX Workplace functions are contained in REXXUTIL.DLL which must be in the LIBPATH.

The function must be loaded before it can be called. (See sample code for SysCreateObject, pg. 25.)

SEE ALSO

SysCreateObject -25, WinOpenObject -16

NOTES

See WinOpenObject (pg. 16) for more information.

● **SysQueryClassList REXX**

Enumerates all registered Workplace classes.

EQUIVALENT C FUNCTION

WinEnumObjectClasses

SYNTAX

result = SysQueryClassList ('List.')

PARAMETERS

List - output

Name of a stem variable. The class list will be placed in the specified stem starting with List.1. List.0 will contain the total number of entries.

RETURNS

1—The call was successful.

0—An error occurred.

OTHER INFO

The REXX Workplace functions are contained in REXXUTIL.DLL which must be in the LIBPATH.

The function must be loaded before it can be called. (See sample code.)

SEE ALSO

SysDeregisterObjectClass -28, SysRegisterObjectClass -33,
WinEnumObjectClasses -14

SAMPLE CODE

```
/* Load the REXX utility function */
call RxFuncAdd 'SysQueryClassList', 'RexxUtil', 'SysQueryClassList'
/* Store the class list in a stem variable called ClassList and
   output to screen*/
result = SysQueryClassList ('ClassList.')
if result = 1 then
    say 'Workplace Classes'
    say '-----'
    do count = 1 to ClassList.0
        say ClassList.count
    else
        say 'SysQueryClassList failed.'
```

● SysRegisterObjectClass REXX

Registers a Workplace class.

EQUIVALENT C FUNCTION

WinRegisterObjectClass

SYNTAX

result = SysRegisterObjectClass (*ClassName*, *ModuleName*)

PARAMETERS

ClassName - input

The name of the class to register. The parameter is case-sensitive, and the name of the class must be unique in the system.

ModuleName - input

The name of the DLL with the class definition.

RETURNS

1—The class registered successfully.

0—An error occurred.

OTHER INFO

The REXX Workplace functions are contained in REXXUTIL.DLL which must be in the LIBPATH.

The function must be loaded before it can be called. (See sample code for SysCreateObject, pg. 25.)

SEE ALSO

SysCreateObject -25, SysDeregisterObjectClass -28,
SysQueryClassList -32, WinRegisterObjectClass -18

NOTES

For more information, see WinRegisterObjectClass (pg. 18).

● SysSaveObject REXX (Warp only)

Notifies an object to save its data.

EQUIVALENT C FUNCTION

WinSaveObject

SYNTAXresult = SysSaveObject (*Object*, *Async*)**PARAMETERS***Object* - input

The object to save. This can be either the object ID (if it exists and is known) or, if the object is a descendant of WPFileSystem, the fully qualified path of the file representing the object.

Async - input

If set to 'True', the object will be saved asynchronously and the function will return immediately. If 'False', the function will not return until after the object has been saved.

RETURNS

1—The object was saved successfully.

0—An error occurred.

OTHER INFO

The REXX Workplace functions are contained in REXXUTIL.DLL which must be in the LIBPATH. The function must be loaded before it can be called. (See sample for SysCreateObject, pg. 25.)

SEE ALSO

SysCreateObject -25, WinSaveObject -21

NOTES

See WinSaveObject (pg. 21) for more information.

● SysSetObjectData REXX

Passes setup information to an object.

EQUIVALENT C FUNCTION

WinSetObjectData

SYNTAX

result = SysSetObjectData (*Object*, *SetupString*)

PARAMETERS

Object - input

The object ID or, if the object is a descendant of WPFileSystem, the fully qualified path of any existing object. Object IDs can be specified in the *SetupString* parameter of SysCreateObject. See Appendix D for a list of object IDs of the predefined Workplace objects.

SetupString - input

The desired settings for the specified object. This parameter is in the same format as the *SetupString* parameter for SysCreateObject. A setup string is a list of keyname value pairs separated by semicolons which is parsed by the object to set instance data. See Appendix D for the list of possible keyname value pairs for the predefined Workplace Shell classes. For more information on setup strings, see wpSetup (pg. 63).

RETURNS

- 1—The setup string was successfully passed to the object.
- 0—The object was not found or an error occurred.

OTHER INFO

The REXX Workplace functions are contained in REXXUTIL.DLL which must be in the LIBPATH.

The function must be loaded before it can be called. (See sample code for SysCreateObject, pg. 25.)

SEE ALSO

SysCreateObject -25, WinSetObjectData -22, wpSetup -63

Installing Workplace Classes: Structures

OBJCLASS

(12 bytes)

pNext (POBJCLASS) 0

is the pointer to the next OBJCLASS node in the list. If pNext is NULL, this node is last in the linked list.

pszClassName (PSZ) 4

is a string containing the name of the class.

pszModName (PSZ) 8

is a string containing the name of the DLL in which this class is defined. This name will only be fully qualified if WinRegisterObjectClass was called with a fully qualified module name for this class.

Used by: WinEnumObjectClasses -14

●3

Object Manipulation

This chapter describes the object manipulation methods that are defined by WPObjct and, therefore, are inherited by all Workplace subclasses. These methods may be called by the system or the programmer to create, move, copy, and delete objects.

Object Dormancy

To conserve memory usage, persistent objects (descendants of either `WPFileSystem` or `WPAbstract`) that are not currently in use either by the user or an application are not loaded in memory. When an object is in memory, it is said to be awake; otherwise, it is dormant. Objects are awakened by the system when they are created or when they come into use and are not yet in memory. Some examples of when an object comes into use are: a view of the object is opened, a view of the parent folder is opened, a shadow of the object is awakened, or the object is a folder and a contained object is awakened. An object can also be awakened by an application with `wpclsQueryObject` or `wpclsObjectFromHandle` (Warp).

The Instance Lock Count

The Workplace Shell maintains a *lock count* for each object in the system to keep track of whether the object is in use or must be made dormant. The system automatically increments the lock count for an object (that is, *locks* the object) each time it is used by the system and decrements the lock count (that is, *unlocks* the object) when it is no longer needed for that particular usage. For example, since an object can have several views open simultaneously, each time a view is opened, the system locks the object, and as each view is closed, the system unlocks the object.

When the lock count for an object is decremented to zero, Workplace puts the object in a list of objects that must be made dormant. Every 90 seconds a thread in Workplace goes through the list and makes each object dormant. This timeout value is configurable in the `CONFIG.SYS` by adding the line

```
SET OBJECTSNOOZETIME = n
```

where *n* is the number of seconds for the timeout. Application developers may want to lower this value on their test machines to force objects to go dormant sooner than 90 seconds. This is advantageous because while objects of a class are awake, the class definition `.DLL` is in use and cannot be replaced with a newer version. One way to replace the `.DLL` is to force all instances to go dormant by closing them and placing them in a closed folder and then waiting for the snooze time to elapse.

Since descendants of `WPTransient` are not persistent, their lock count is ignored and they are never made dormant. To release a transient object from memory, call `wpFree` to destroy it. If an object were to go dormant while an application is manipulating the object (calling methods on the object), a trap in Workplace would occur. To avoid this, all methods that return an object pointer will lock the object before returning, either automatically or at the programmer's discretion through the use of an `fLock` parameter. It is the responsibility of the programmer to call `wpUnlockObject` when the object is no longer needed by the application. In Warp, the programmer can use `wpLockObject` to place additional locks on an object to ensure that it will remain awake.

Restrictions and Warnings

- It is not guaranteed that a method will be called whenever an object is created, deleted, or moved. Sometimes, Workplace will manipulate an object without calling these methods. For example, if a folder is copied or moved, public methods are not called to copy or move its contents.
- Workplace uses a dialog for copy, move, create another, and create shadow actions selected from the pop-up menu. There is no programming interface to use or modify these dialogs.
- The method `wpLockObject` was not made public until Warp. OS/2 2.1 applications can lock an object by first calling `wpQueryHandle` and then `wpclsQueryObject`.
- Be aware that dereferencing a SOM pointer for an object that has gone dormant will cause a trap. Methods that lock objects will be documented as such in this reference. Until you unlock the object, it is safe to access the SOM pointer. If you pass an object pointer to a function that runs asynchronously, be sure that you lock the object or that you only access the SOM pointer when you know the object must still be awake.
- In version 2.1, if `wpCreateFromTemplate` is called on a template object with an object ID, it will copy the object ID to the new object. This is bad because object IDs must be unique among objects; this bug was fixed in Warp. The workaround in 2.1 would be to avoid placing object IDs on template objects or to override `wpCopiedFromTemplate` to reset the object ID to a blank string (by calling `wpSetup` on the new object and passing "OBJECT=" as the setup string).

Instance Methods

wpConfirmDelete prompts the user for confirmation on the deletion of an object (pg. 40).

wpCopiedFromTemplate notifies an object that it has been copied from a template (pg. 43).

wpCopyObject copies an object to the specified folder (pg. 44).

wpCreateAnother creates another object of the same class (pg. 47).

wpCreateFromTemplate creates a new object from a template (pg. 48).

wpCreateShadowObject creates a shadow of an object in the specified folder (pg. 50).

wpDelete deletes an object using the specified confirmations (pg. 51).

wpFree deletes an object regardless of the object's style (pg. 53).

wpIsDeleteable (Warp) tests if an object can be deleted (pg. 54).

wpIsLocked (Warp) tests if an object is locked (pg. 55).

wpIsObjectInitialized (Warp) tests if an object is fully initialized (pg. 56).

wpLockObject (Warp) increments an object's lock count by one (pg. 56).

wpMoveObject moves an object to the specified folder (pg. 57).

wpObjectReady (Warp) notifies an object when it becomes completely initialized (pg. 58).

wpQueryHandle returns the handle of an object (pg. 60).

wpScanSetupString parses a setup string (pg. 61).

wpSetup sets or changes the settings of an object (pg. 63).

wpSetupOnce (Warp) sets the settings of an object when it is first created (pg. 64).

wpUnlockObject reduces an object's lock count by one (pg. 65).

● **wpConfirmDelete** **WPObject instance method**

Prompts the user for confirmation on the deletion of an object.

SYNTAX

ULONG wpConfirmDelete (**ULONG** *fConfirmations*)

PARAMETERS

fConfirmations - input

Any combination of the following flags ORed together:

Constant		Defined Name
CONFIRM_DELETE	0x01	Confirm deletion of all objects.
CONFIRM_DELETEFOLDER	0x02	Confirm deletion of folders.

RETURNS

Define		Defined Name
OK_DELETE	1	The object can be deleted.
NO_DELETE	2	The object cannot be deleted.
CANCEL_DELETE	3	The user canceled the entire delete operation.

HOW TO CALL

Call this method on any object to confirm its deletion. The user will be prompted with a message box asking for confirmation to delete an object or an entire folder, depending upon the confirmation flags specified.

HOW TO OVERRIDE

Override this method to change its behavior. Prompt the user for confirmations and return one of the valid *_DELETE codes (see above) without calling the parent method, or call the parent method to have the system put up the message box confirming the deletion.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpDelete -51, wpFree -53, wpIsDeleteable -54,
wpQueryConfirmations -111

NOTES

If you are deleting many objects and wpConfirmDelete returns CANCEL_DELETE, cancel the entire delete operation and do not delete any more objects. If NO_DELETE is returned, do not delete the one object but continue with the others.

RESTRICTIONS/WARNINGS

- If you are overriding `wpConfirmDelete`, note that the *fConfirmations* parameter could contain other flags that are unrelated to deletion. Do not test *fConfirmations* for zero to determine if `CONFIRM_DELETE` and/or `CONFIRM_DELETEFOLDER` are present.
- When the system deletes an object as a result of the user selecting “Delete” from the pop-up menu, a special confirmation dialog is presented to the user, and `wpDelete` is called on each object with *fConfirmations* set to zero. In this case, `wpConfirmDelete` is *not* called. However, when the system deletes an object as a result of the user pressing the Delete key on the keyboard, `wpDelete` is called with *fConfirmations* set to `CONFIRM_DELETEFOLDER` if the object is a folder, and `CONFIRM_DELETE` otherwise. In this case, `wpConfirmDelete` is called.

SAMPLE CODE

The following sample obtains an object pointer for a folder using its object ID and deletes it using `wpConfirmDelete`. The context of this sample could be within a method or a regular function.

```
SOMAny *Object;
ULONG ulAnswer;
BOOL bSuccess = FALSE;
/* Wake up a folder with the object ID, <MY_FOLDER> and delete it */
Object = _wpclsQueryFolder(_WPObject, "<MY_FOLDER>", TRUE);
if (Object)
{
    /* Confirm deletion from the user but pass the default
       confirmation flags with wpQueryConfirmations to get the user
       delete confirmation preferences from the system object.
       wpConfirmDelete will only look at the flags that involve
       object deletion.
    */
    ulAnswer = _wpConfirmDelete(Object, _wpQueryConfirmations(Object));
    if (ulAnswer == OK_DELETE)
    {
        /* we can call wpFree instead of wpDelete because we already
           handled the
        */
        * confirmations.
    }
}
```



```
        bSuccess = _wpFree(Object);
    }
    if (!bSuccess)
    {
        /* the deletion was not successful. Unlock the object so it can
           go dormant*/
        _wpUnlockObject(Object);
    }
}
```

● **wpCopiedFromTemplate** **WPObj** instance method

Notifies an object that has been copied from a template.

SYNTAX

VOID wpCopiedFromTemplate ()

PARAMETERS

none

RETURNS

none

HOWTO CALL

This method is generally called only by the system on a new object after it has been copied from a template.

HOWTO OVERRIDE

Override this method to perform any necessary processing or initialization once a new object has been copied from a template.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpclsNew -71, wpclsQueryStyle -127, wpCopyObject -44,
wpCreateFromTemplate -48, wpModifyStyle -108, wpObjectReady -58

NOTES

An object is copied from a template when the user drags from a template object or selects a template name from the Create Another pull-down in a pop-up menu. Workplace Shell calls the `wpCreateFromTemplate` method to create the new object and then on the new object for notification. Neither `wpCopyObject` nor `wpclsNew` is called. The new object is an exact copy of the template, except that the `OBJSTYLE_TEMPLATE` and `OBJSTYLE_NODELETE` styles are removed.

RESTRICTIONS/WARNINGS

- If your class style is `CLSSTYLE_NEVERDELETE`, `wpCreateFromTemplate` will still remove the `OBJSTYLE_NODELETE` flag from the new object. Override `wpCopiedFromTemplate` to put the `OBJSTYLE_NODELETE` style back on the new object (see sample below).
- `wpObjectReady` (Warp) can also be overridden for notification when an object has been created from a template.

SAMPLE CODE

The following sample is an override of the `wpCreateFromTemplate` method for a class called `ProtectedClass`. The default style of this class includes `CLSSTYLE_NEVERDELETE`. `wpCreateFromTemplate` is overridden to put the `OBJSTYLE_NODELETE` flag on the new object.

```
SOM_Scope void SOMLINK pcls_wpCopiedFromTemplate
(ProtectedClass *somSelf)
{
    /* put the OBJSTYLE_NODELETE style back on the object */
    _wpModifyStyle(somSelf, OBJSTYLE_NODELETE, OBJSTYLE_NODELETE);
    /* mark the object to save its data */
    _wpSaveDeferred(somSelf);
    parent_wpCopiedFromTemplate(somSelf);
}
```

● **wpCopyObject** **WPObject** instance method

Copies an object to the specified folder.

SYNTAX

WPObject * `wpCopyObject` (**WPFolder** **Folder*, **BOOL** *fLock*)

PARAMETERS

Folder—input

The object pointer of the folder in which to place the newly copied object. Some methods for obtaining Folder object pointers are `wpclsQueryFolder`, `wpclsQueryObject`, and `wpclsQueryObjectFromPath`.

fLock—input

TRUE—Increment the lock count on the new object returned.

FALSE—Do not increment the lock count on the new object.

RETURNS

(nonzero value)—The object pointer for the new copy of the object.

NULL—An error occurred.

HOW TO CALL

Call this method on any object to make an exact copy. The storage class will copy all the persistent instance data of the original and save it in the copy. The object ID and object handle of the original, if they exist, will not be copied to the new object.

HOW TO OVERRIDE

Override this method for notification when an object is being copied. Call the parent method to do the actual copy and be sure to return the new object from your override. If the parent method is not called, the object will not be copied properly. See the sample code for an example of overriding this method.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpCreateFromTemplate` -48, `wpCreateAnother` -47, `wpclsNew` -71, `wpclsObjectFromHandle` -72, `wpclsQueryFolder` -74, `wpclsQueryObject` -75, `wpclsQueryObjectFromPath` -76, `wpMoveObject` -57, `wpObjectReady` -58, `wpUnlockObject` -65

RESTRICTIONS/WARNINGS

- Pass TRUE for *fLock* if you are going to use the returned object pointer when calling `wpCopyObject`. Otherwise, the object might be freed before you are finished with the pointer. Be sure to call `wpUnlockObject` when the pointer is no longer needed.

- If you are overriding the method to manipulate the new copy and the caller specified FALSE for fLock, pass TRUE to the parent method and then call wpUnlockObject on the new object when you are finished (see sample below).
- When an object is copied, the storage classes use the save and restore methods to copy the data from one object to another. This means that any data that is not saved and restored in the wpSaveState and wpRestoreState methods will not get copied. In the case of WPTransient descendants, no data will be copied because WPTransient does not process the save and restore methods. Override wpCopyObject to copy any data that would not be copied by the storage class.
- If there is persistent instance data defined by your class that is unique for each object or would not make sense to be copied over to the new object, override wpCopyObject and change the data after calling the parent method (see sample below).
- wpCopyObject is not always called on an object when it is copied by the system. If a folder is copied, wpCopyObject will not be called on the folder's contents.
- wpObjectReady (Warp) can also be overridden for notification when an object is copied.

SAMPLE CODE

The following sample demonstrates how to override wpCopyObject to perform an action on the new copy. In this case, the action will be to call the method SetUniqueData on the object to reset a persistent instance variable to zero. (SetUniqueData is assumed to be defined for this class and is not shown here.) The name of the class in this example is MyClass.

```
SOM_Scope WPObject * SOMLINK mycls_wpCopyObject(MyClass *somSelf,
                                                WPFolder *Folder,
                                                BOOL fLock)
{
    SOMAny *NewObject;
    /* First call the parent method to perform the copy. Pass TRUE for
       fLock */
    NewObject = parent_wpCopyObject(somSelf, Folder, TRUE);
    /* reset the data */
    _SetUniqueData(NewObject, 0);
}
```

```

/* mark the object to save its data */
_wpSaveDeferred(NewObject);
/* if the caller had not specified to lock the object, unlock it
*/
if (!fLock)
    _wpUnlockObject(NewObject);
return(NewObject);
}

```

● **wpCreateAnother** **WPObject** instance method

Creates another object of the same class.

SYNTAX

WPObject * wpCreateAnother (**PSZ** *pszTitle*, **PSZ** *pszSetupEnv*,
 WPFolder **Folder*)

PARAMETERS

pszTitle - input

The title for the new object.

pszSetupEnv - input

The initial settings for the new object. A setup string is a list of key-name value pairs separated by semicolons which is parsed by the object for initialization. Specify NULL to use the object's defaults. See Appendix A for the list of possible keyname value pairs for the predefined Workplace Shell classes. For more information on setup strings, see wpSetup (pg. 63).

Folder - input

The object pointer of the folder in which to place the newly created object. Some methods for obtaining Folder object pointers are wpclsQueryFolder, wpclsQueryObject, and wpclsQueryObjectFromPath.

RETURNS

(nonzero value)—The object pointer for the new object.

NULL—An error occurred.

HOW TO CALL

Call this method on any object to create a new object of the same class.

HOW TO OVERRIDE

Override this method for notification when the user chooses the default Create another pop-up menu item from an object.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpclsNew` -71, `wpclsObjectFromHandle` -72, `wpclsQueryFolder` -74, `wpclsQueryObject` -75, `wpclsQueryObjectFromPath` -76, `wpObjectReady` -58, `wpSetup` -63, `wpUnlockObject` -65

RESTRICTIONS/WARNINGS

- The `WObject` class handles this method by querying the class of the object and calling `wpclsNew` with the same parameters. The *fLock* parameter is passed as `TRUE` to `wpclsNew`, therefore the returned new object will be locked and must be unlocked with `wpUnlockObject` when its pointer is no longer needed.
- This method is predominantly useful to override for notification. Calling `wpclsNew` directly is more flexible and can be used instead to create new objects.
- `wpCreateAnother` is only called by the system when the user chooses the default Create another item in the pop-up menu of an object. `wpclsNew` is called directly when `WinCreateObject` is called from an application or when the system creates objects.
- `wpObjectReady` (*Warp*) can be overridden instead for notification of creation of a new object.

-
- | | |
|-------------------------------|-----------------------------------|
| ● wpCreateFromTemplate | WObject
instance method |
|-------------------------------|-----------------------------------|
-

Creates a new object from a template.

SYNTAX

`WObject * wpCreateFromTemplate (WPFolder *folder, BOOL fLock)`

PARAMETERS

folder - input

The object pointer of the folder in which to place the newly created object. Some methods for obtaining folder object pointers are `wpclsQueryFolder`, `wpclsQueryObject`, and `wpclsQueryObjectFromPath`.

fLock - input

TRUE—Increment the lock count on the new object returned.

FALSE—Do not increment the lock count on the new object.

RETURNS

(nonzero value)—The object pointer for the new object.

NULL—An error occurred.

HOW TO CALL

Call this method on any template to create a new object. A template is an object with the style `OBJSTYLE_TEMPLATE`.

HOW TO OVERRIDE

This method does not need to be overridden for notification purposes because `wpCopiedFromTemplate` and `wpObjectReady` (Warp) will be invoked on the newly created object. The returned object will have the same settings as the original template except the `OBJSTYLE_NO_DELETE`, if present, and `OBJSTYLE_TEMPLATE` styles are removed.

OTHER INFO

Include file: `wobject.h`

SEE ALSO

`wpclsNew` -71, `wpclsQueryFolder` -74, `wpclsQueryObject` -75,
`wpclsQueryObjectFromPath` -76, `wpCopiedFromTemplate` -43,
`wpModifyStyle` -108, `wpObjectReady` -58, `wpSetup` -63,
`wpUnlockObject` -65

NOTES

An object is copied from a template when the user drags from a template object or selects a template name from the Create another pulldown in a pop-up menu. Workplace Shell calls the `wpCreateFromTemplate` method to create the new object and then `wpCopiedFromTemplate` on the new object for notification. Neither `wpCopyObject` nor `wpclsNew` is called. The new object is an exact copy of the

template except the `OBJSTYLE_TEMPLATE` and `OBJSTYLE_NODELETE` styles are removed.

RESTRICTIONS/WARNINGS

- If your class style is `CLSSTYLE_NEVERDELETE`, `wpCreateFromTemplate` will still remove the `OBJSTYLE_NODELETE` flag from the new object. Override `wpCreateFromTemplate` or `wpObjectReady` (Warp) to put the `OBJSTYLE_NODELETE` style back on the new object.
- Pass `TRUE` for the *fLock* parameter if you are going to access the object pointer after calling `wpCreateFromTemplate`. When the pointer is no longer needed, call `wpUnlockObject`.
- In version 2.1, if `wpCreateFromTemplate` is called on a template object with an object ID, it will copy the object ID to the new object. This is bad because object IDs must be unique among objects; this bug was fixed in Warp. The workaround in 2.1 would be to avoid placing object IDs on template objects or to override `wpCopiedFromTemplate` to reset the object ID to a blank string (by calling `wpSetup` on the new object and passing "OBJECT=" as the setup string).

● wpCreateShadowObject	WPObject instance method
-------------------------------	------------------------------------

Creates a shadow of an object in the specified folder.

SYNTAX

WPObject * `wpCreateShadowObject` (**WPFolder** **Folder*, **BOOL** *fLock*)

PARAMETERS

Folder - input

The object pointer of the folder in which to place the shadow. Some methods for obtaining Folder object pointers are `wpclsQueryFolder`, `wpclsQueryObject`, and `wpclsQueryObjectFromPath`.

fLock - input

`TRUE`—Increment the lock count on the new shadow object returned.

`FALSE`—Do not increment the lock count on the new shadow object.

RETURNS

(nonzero value)—The object pointer for the new object.
NULL—An error occurred.

HOW TO CALL

Call this method on any object to create a shadow of the object. A `USAGE_LINK` item will be added to the in-use list of the original object. See Chapter 15 for more information on shadows and Chapter 6 for more information on in-use lists.

HOW TO OVERRIDE

This method can be overridden for notification when a user creates a shadow of an object by using `Ctrl+Shift+Drag` or selecting `Create shadow` from the pop-up menu.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpAddToObjUseList` -145, `wpclsQueryFolder` -74,
`wpclsQueryObject` -75, `wpclsQueryObjectFromPath` -76,
`wpQueryShadowedObject` -298, `wpUnlockObject` -65

NOTES

Pass `TRUE` for the *fLock* parameter if you are going to access the shadow object pointer after calling `wpCreateShadowObject`. When the pointer is no longer needed, call `wpUnlockObject`.

● **wpDelete** **WPObj** instance method

Deletes an object using the specified confirmations.

SYNTAX

ULONG wpDelete (**ULONG** *fConfirmations*)

PARAMETERS

fConfirmations - input

Specify 0 for no user confirmation or any combination of the following flags ORed together:

Constant		Description
CONFIRM_DELETE	0x01	Confirm deletion of all objects.
CONFIRM_DELETEFOLDER	0x02	Confirm deletion of folders.

RETURNS

Constant		Description
OK_DELEE	1	The object was deleted.
NO_DELETE	2	An error occurred or the user specified No at the prompt.
CANCEL_DELETE	3	The user canceled the delete operation.

HOW TO CALL

Call this method on any object to delete it. The WPObject class will first make sure the object does not have the style OBJSTYLE_NODELETE. Then it calls wpConfirmDelete if delete confirmation flags are passed in. If wpConfirmDelete returns OK_DELETE, wpFree is called to delete the object.

HOW TO OVERRIDE

This method can be overridden to do special processing when an object is deleted, but wpDelete is not always called by the system to delete an object. It is better to override wpFree for this purpose.

OTHER INFO

Include file: wobject.h

SEE ALSO

WinDestroyObject -13, wpConfirmDelete -40, wpDeleteContents -236, wpFree -53, wpIsDeleteable -54, wpQueryConfirmations -111

RESTRICTIONS/WARNINGS

- Before calling wpDelete, you should call wpQueryConfirmations on the object to determine which flags to pass in the *fConfirmations* parameter.
- Do not call wpDelete on the somSelf pointer from within an instance method override. The object and its SOM pointer will be destroyed before the method returns. Any code that subsequently accesses that

pointer either from the override itself or from the caller of the method will trap. One way an object can delete itself would be to post a message to an application-defined thread to do the deletion.

- **WPFolder** overrides **wpDelete** to delete its contents with **wpDeleteContents** before calling the parent method to delete the folder object itself.
- If **wpDelete** is called on a folder containing many objects, it should be done from a separate thread to avoid temporarily hanging the user interface.
- When the system deletes an object as a result of the user selecting **Delete** from the pop-up menu, a special confirmation dialog is presented to the user, and **wpDelete** is called on each object with *fConfirmations* set to zero. In this case, **wpConfirmDelete** is *not* called. However, when the system deletes an object as a result of the user pressing the **Delete** key on the keyboard, **wpDelete** is called with *fConfirmations* set to **CONFIRM_DELETEFOLDER** if the object is a folder, and **CONFIRM_DELETE** otherwise. In this case, **wpConfirmDelete** is called.

● **wpFree** **WObject** instance method

Deletes an object.

SYNTAX

BOOL wpFree ()

PARAMETERS

none

RETURNS

TRUE—The object was successfully deleted.

FALSE—An error occurred.

HOW TO CALL

Call this method on any object to delete it. If user confirmation from the user is desired, call the **wpDelete** method instead.

HOW TO OVERRIDE

This method can be overridden to do special processing when an object is deleted.

OTHER INFO

Include file: wobject.h

SEE ALSO

WinDestroyObject -12, wpConfirmDelete -40, wpDelete -51,
wpDeleteContents -236, wpIsDeleteable -54,
wpQueryConfirmations -111

RESTRICTIONS/WARNINGS

- Do not call wpFree on the somSelf pointer from within an instance method override. The object and its SOM pointer will be destroyed before the method returns. Any subsequent code that accesses that pointer either from the override itself or from the caller of the method will trap. One way an object can delete itself would be to post a message to an application-defined thread to do the deletion.
- If wpFree is called on a folder containing many objects, it should be done from a separate thread to avoid temporarily hanging the user interface.
- WPFolder overrides wpFree and deletes its contents with wpDeleteContents before calling the parent method to delete the folder object itself.

● **wpIsDeleteable** **WPObject instance method**
 (Warp only)

Tests if an object can be deleted.

SYNTAX

BOOL wpIsDeleteable ()

PARAMETERS

none

RETURNS

TRUE—The object can be deleted.

FALSE—The object cannot be deleted.

HOW TO CALL

Call this method on any object to determine if it can be deleted by the user. Although any object can be deleted using wpFree, this method is useful in determining if an object is meant to be deleted.

HOWTO OVERRIDE

This method can be overridden to do special processing to determine if an object can be deleted.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpConfirmDelete` -40, `wpDelete` -51, `wpDeleteContents` -236, `wpFree` -53, `wpQueryConfirmations` -111

● **wpIsLocked** **WPyObject instance method (Warp only)**

Tests if an object is locked.

SYNTAX

BOOL `wpIsLocked` ()

PARAMETERS

none

RETURNS

TRUE—The object is locked.

FALSE—The object is not locked.

HOWTO CALL

Call this method on any object to determine if it is locked. If an object's lock count is greater than zero, it is considered locked and cannot be made dormant.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpLockObject` -56, `wpUnlockObject` -65

PARAMETERS

none

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method on any object to increment its lock count by one. As long as the lock count of an object is nonzero, it will not be made dormant by the system. If an object is referenced in asynchronous processing, call `wpLockObject` before using the object and then `wpUnlockObject` when it is no longer needed.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.`

SEE ALSO

`wpUnlockObject` -65

NOTES

The lock count is an internal counter the `WPObj` class defines to keep track of whether an object is needed in memory. The system will increment the lock count of an object once for every open view in the in-use list, once for every open folder view in which the object is contained, once for every shadow of the object, once for every contained object if the object is a folder, and once every time a method is called with the *fLock* parameter set to TRUE. It is the application's responsibility to call `wpUnlockObject` after locking an object.

● **wpMoveObject** **WPObj instance method**

Moves an object to the specified folder.

SYNTAX

BOOL wpMoveObject (**WPFolder** *Folder)

PARAMETERS

Folder - input

The pointer of the new folder location for the object. Some methods for obtaining Folder object pointers are `wpclsQueryFolder`, `wpclsQueryObject`, and `wpclsQueryObjectFromPath`.

RETURNS

TRUE—The object was successfully moved.

FALSE—An error occurred.

HOW TO CALL

Call this method on any object to move it to another folder.

HOW TO OVERRIDE

Override this method for notification when an object is being moved. Call the parent method to do the actual move. If the parent method is not called, the object will not be moved properly.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpclsQueryFolder` -74, `wpclsQueryObject` -75,
`wpclsQueryObjectFromPath` -76, `wpCopyObject` -44

RESTRICTIONS/WARNINGS

- `wpMoveObject` is not always called on an object when it is moved by the system. If a folder is moved, `wpMoveObject` will not be called on the folder's contents.

● **`wpObjectReady` `WPObject` instance method
(Warp only)**

Notifies an object when it becomes completely initialized.

SYNTAX

VOID `wpObjectReady` (**ULONG** *ulCode*, **WPObject** * *refObject*)

PARAMETERS

ulCode - input

The following flags are ORed together to provide information about how the object was instantiated:

Constant		Description
OR_NEW	0x01	This is a new object that has been created.
OR_AWAKE	0x02	This object already existed and has just been awakened.
OR_REFERENCE	0x10000000	The <i>refObject</i> parameter has been set.
OR_FROMTEMPLATE	0x10000004	This object was created from the template <i>refObject</i> .
OR_FROMCOPY	0x10000008	This object was copied from the object <i>refObject</i> .
OR_SHADOW	0x10000010	This object was just created as a shadow of <i>refObject</i> .

refObject - input

The pointer to the reference object this object was created from if *ulCode* contains OR_REFERENCE. Otherwise it is set to NULL.

RETURNS

none

HOW TO CALL

This method is generally called only by the system.

HOW TO OVERRIDE

Override this method for notification when an object has become fully initialized after its instantiation.

OTHER INFO

Include file: `wobject.h`

SEE ALSO

`wpIsObjectInitialized` -56, `wpSetup` -63, `wpSetupOnce` -64

● **wpQueryHandle** **WPObj** instance method

Returns the handle of an object.

SYNTAX

HOBJECT wpQueryHandle ()

PARAMETERS

none

RETURNS

(nonzero value)—The persistent handle of the object.

NULLHANDLE—An error occurred.

HOW TO CALL

Call this method on any object to retrieve its persistent handle. This handle can be used at any time to get the object pointer using wpclsQueryObject or wpclsObjectFromHandle (Warp).

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wpobject.h

SEE ALSO

WinQueryObject -17, wpclsObjectFromHandle -72,
wpclsQueryObject -75, WinCreateObject -9, WinDestroyObject -13,
WinSetObjectData -22

NOTES

Workplace objects, other than transients, are not always loaded in memory. If an object must be accessed but it might not be awake and in memory, the persistent object handle can be used with the wpclsQueryObject method to wake up the object. wpclsQueryObject can also be used if the object is already awake but the SOM pointer is not known. Object handles are also useful for non-SOM applications on a separate process that manipulate Workplace Shell objects. A Workplace object can query its handle with wpQueryHandle and then pass it to another process. The other process could then call WinSetObjectData or WinDestroyObject with this handle.

RESTRICTIONS/WARNINGS

- File system objects are not assigned handles until `wpQueryHandle` is called. The handles for these objects are stored in a table in the `os2sys.ini` file that is also kept in memory. `wpQueryHandle` should be used sparingly on file system objects because if it were called on all files on the system, the large size of the table would greatly affect system performance.

● **wpScanSetupString** **WPObj** instance method

Parses a setup string.

SYNTAX

BOOL wpScanSetupString (**PSZ** *pszSetupString*, **PSZ** *pszKey*, **PSZ** *pszValue*, **PULONG** *pcbValue*)

PARAMETERS

pszSetupString - input

List of keyname value pairs separated by semicolons. Setup strings are passed to objects via the `wpSetup` method.

pszKey - input

The keyname that represents the data that is to be extracted from the setup string. See Appendix A for the list of possible keyname value pairs for the predefined Workplace Shell classes.

pszValue - input/output

The buffer to place the data extracted from the setup string. If `NULL` is specified, the required size for the buffer is returned in *pcbValue*.

pcbValue - input/output

If *pszValue* is nonzero, set *pcbValue* to point to the size of *pszValue*. If *pszValue* is `NULL`, the required size for the buffer is returned in this parameter.

RETURNS

TRUE—The specified keyname was found and if *pszValue* was non-null, the data was copied to the buffer.

FALSE—An error occurred or the keyname was not found.

HOWTO CALL

Call this method on any object to parse a setup string. This method is usually called from an override of the `wpSetup` method to extract setup data.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`WinSetObjectData` -22, `WinCreateObject` -9, `wpclsNew` -71, `wpSetup` -63

SAMPLE CODE

A setup string is a list of keyname value pairs separated by semicolons. For example, the setup string, "NODELETE=YES;NOMOVE=YES;OBJECTID=<MY_OBJECTID>" would be used to set up an object such that its object ID is <MY_OBJECTID>, it is not deleteable, and it is not moveable. Classes can define their own keyname value pairs to allow instance data or other values to be set via `wpSetup`. The following sample is an override of the `wpSetup` method from the source of a class called `MyClass`.

```
SOM_Scope BOOL  SOMLINK mcls_wpSetup(MyClass *somSelf,
                                     PSZ pszSetupString)
{
    ULONG cbValue;
    BOOL bReturn;
    MyClassData *somThis = MyClassGetData(somSelf);
    cbValue = 0;
    /* find the value "NAME" in the setup string and set the instance
       data _pszName on this object
    */
    bReturn = _wpScanSetupString(somSelf, pszSetupString, "NAME",
                                NULL, &cbValue);

    if (bReturn && cbValue)
    {
        _pszName = (PSZ) _wpAllocMem(somSelf, cbValue, NULL);
        if (_pszName)
```

```
        _wpScanSetupString(somSelf, pszSetupString, "NAME",  
                           _pszName, &cbValue);  
    }  
    return parent_wpSetup(somSelf, pszSetupString);  
}
```

● **wpSetup** **WPObj** instance method

Sets or changes the settings of an object.

SYNTAX

BOOL wpSetup (**PSZ** *pszSetupString*)

PARAMETERS

pszSetupString - input

List of keyname value pairs separated by semicolons. See Appendix A for the list of possible keyname value pairs for the predefined Workplace Shell classes.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method on any object to pass it a setup string to set data. wpSetup does not need to be called just after creating an object; wpclsNew and WinCreateObject both have setup string parameters that will be passed to the object when the system calls wpSetup.

HOW TO OVERRIDE

Override this method to extract the class-defined data from the setup string, using wpScanSetupString to parse the string.

OTHER INFO

Include file: wpobject.h

SEE ALSO

WinSetObjectData -22, WinCreateObject -9, wpclsNew -71,
wpRestoreState -89, wpSaveState -97, wpScanSetupString -61,
wpSetupOnce -64

NOTES

A setup string is a list of keyname value pairs separated by semicolons. For example, the setup string “NODELETE=YES;NOMOVE=YES;OBJECTID=<MY_OBJECTID>” would be used to set up an object such that its object ID is <MY_OBJECTID>, it is not deleteable, and it is not moveable. Classes can define their own keyname value pairs to allow instance data or other values to be set via wpSetup. If it is necessary to have a literal semicolon as part of the data, put a carat (^) before the semicolon; otherwise it will be treated as a separator between keyname value pairs. Whenever possible, use commas to separate multiple values for one keyname.

RESTRICTIONS/WARNINGS

- wpSetup is called when the object is first created and could be called again by any application or as a result of any application calling WinSetObjectData or SysSetObjectData. If a setup keyname must be processed only once (when the object has just been created), wpSetupOnce (Warp) should be overridden to handle this.
- The system will always call wpSaveDeferred on an object immediately after it calls wpSetup. Therefore, it is not necessary to make the call to wpSaveDeferred from within a wpSetup *override*. However, wpSaveDeferred should be called by any application after *calling* wpSetup.

● **wpSetupOnce WPObject instance method (Warp only)**

Sets or changes the settings of an object when it is first created.

SYNTAX

BOOL wpSetupOnce (**PSZ** *pszSetupString*)

PARAMETERS

pszSetupString - input

List of keyname value pairs separated by semicolons. See Appendix A for the list of possible keyname value pairs for the predefined Workplace Shell classes.

RETURNS

TRUE—The method completed successfully.
FALSE—An error occurred.

HOW TO CALL

This method is called exactly once by the system for each object, namely when the object is created. Do not call this method directly.

HOW TO OVERRIDE

Override this method to extract the class-defined data from the setup string, using `wpScanSetupString` to parse the string. The parent method must be called, which will invoke the `wpSetup` method with the same setup string. Therefore, do not process any setup keynames that are already processed in the `wpSetup` override. Any other one-time setup for an object should be done from within this override as well.

OTHER INFO

Include file: `wobject.h`

SEE ALSO

`WinSetObjectData` -22, `WinCreateObject` -9, `wpclsNew` -71, `wpRestoreState` -89, `wpSaveState` -97, `wpScanSetupString` -61, `wpSetup` -63

NOTES

A setup string is a list of keyname value pairs separated by semicolons. For example, the setup string "NODELETE=YES;NOMOVE=YES;OBJECTID=<MY_OBJECTID>" would be used to set up an object such that its object ID is <MY_OBJECTID>, it is not deletable, and it is not moveable. Classes can define their own keyname value pairs to allow instance data or other values to be set via `wpSetup`. If it is necessary to have a literal semicolon as part of the data, put a carat (^) before the semicolon; otherwise it will be treated as a separator between keyname value pairs. Whenever possible, use commas to separate multiple values for one keyname.

RESTRICTIONS/WARNINGS

- It is not necessary to call `wpSaveDeferred` from within the `wpSetupOnce` method. The system will call `wpSaveDeferred` on the object after `wpSetupOnce` has been invoked.

● **wpUnlockObject** **WPObj** instance method

Reduces an object's lock count by one.

SYNTAX

BOOL wpUnlockObject ()

PARAMETERS

none

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method on any object to decrement its lock count by one. When the lock count reaches zero, the object will go dormant. This method has no effect on WPTransient descendants.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpAddToObjUseList -145, wpclsNew -71, wpclsQueryObject -75, wpCopyObject -44, wpCreateAnother -47, wpCreateFromTemplate -48, wpCreateShadowObject -50, wpIsLocked -55, wpLockObject -56, wpPopulate -242, wpQueryContent -243, wpQueryHandle -60

NOTES

The lock count is an internal counter the WPObj class defines to keep track of whether an object is needed in memory. The system will increment the lock count of an object once for every open view in the in-use list, once for every open folder view the object is contained in, once for every shadow of the object, once for every contained object if the object is a folder, and once every time a method is called with the *fLock* parameter set to **TRUE**. It is the application's responsibility to call wpUnlockObject after locking an object.

RESTRICTIONS/WARNINGS

- If you call the wpPopulate method on a folder, the system will lock all the folder's contents once. You must call wpUnlockObject on

each object contained in the folder when they are no longer needed (use `wpQueryContent` to enumerate the contents of the folder).

- `wpLockObject` was not externalized before Warp. A workaround for locking an object would be to call `wpQueryHandle` to get the object's handle and then `wpclsQueryObject` on the returned handle. Since the object is already awake, the only effect `wpclsQueryObject` will have is to lock the object. Be aware that `wpQueryHandle` use on file system objects should be done in moderation.

Class Methods

`wpclsCreateDefaultTemplates` creates class templates in the templates folder (pg. 67).

`wpclsDecUsage` (Warp) decrements the usage count of a class by one (pg. 69).

`wpclsIncUsage` (Warp) increments the usage count of a class by one (pg. 70).

`wpclsNew` creates a new instance of a class (pg. 71).

`wpclsObjectFromHandle` (Warp) returns the SOM pointer for an object, given its handle (pg. 72).

`wpclsQueryAwakeObject` (Warp) tests if a file system object is awake (pg. 73).

`wpclsQueryFolder` returns the SOM pointer for a folder, given its location (pg. 74).

`wpclsQueryObject` returns the SOM pointer for an object, given its handle (pg. 75).

`wpclsQueryObjectFromPath` returns the SOM pointer for a file system object, given its location (pg. 76).

● **`wpclsCreateDefaultTemplates`** **WPObject class method**

Creates class templates in the templates folder.

SYNTAX

BOOL `wpclsCreateDefaultTemplates` (**WPFolder** * *Folder*)

PARAMETERS

Folder - input

The object pointer for the folder in which to place the templates.

RETURNS

TRUE—Templates have been created.

FALSE—No templates have been created.

HOW TO CALL

This method is generally called only by the Templates folder, when it is populated, on any class that does not have the CLSSTYLE_NEVERTEMPLATE style. If the method returns TRUE, the Templates folder takes no action. If it returns FALSE, the Templates folder will create one nondeleteable template for the class.

HOW TO OVERRIDE

Override this method to create one or more templates in the Templates folder and return TRUE if successful. Do not call the parent method or the parent classes might create duplicate templates. WPObjct takes no action for this method and only returns FALSE.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpcsNew -71, wpcsQueryStyle -127

RESTRICTIONS/WARNINGS

- If your subclass should never have template instances, override wpcsQueryStyle and OR the CLSSTYLE_NEVERTEMPLATE style with the return code from the parent method.
- To prevent the Templates folder from creating templates of your class, override wpcsCreateDefaultTemplates and return TRUE.

SAMPLE CODE

The following sample is an override of the wpcsCreateDefaultTemplates method by the class called MyClass.

```
SOM_Scope BOOL  SOMLINK mclsM_wpcsCreateDefaultTemplates
                                     (M_MyClass *somSelf,
                                     WPFolder *Folder)
```

```
{
    MyClass *NewTemplate;
    /* note somSelf is a class pointer in this override so it can be
    passed to wpclsNew.
    */
    NewTemplate = _wpclsNew(somSelf,_wpclsQueryTitle(somSelf),
                           "TEMPLATE=YES",
                           Folder,
                           FALSE);

    if (NewTemplate)
        return TRUE;
    else
        return FALSE;
}
```

● **wpclsDecUsage** **WPObject class method (Warp only)**

Decrements the usage count of a class by one.

SYNTAX

VOID wpclsDecUsage ()

PARAMETERS

none

RETURNS

none

HOWTO CALL

Call this method on a class object pointer to decrement its usage count once. Only call this method if wpclsIncUsage was previously called.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpclsIncUsage -70

NOTES

The system uses the class usage count to determine whether the class should be unloaded. The usage count is automatically incremented by the system when an instance is created and decremented when an instance goes dormant. Once the count goes back to zero, the class DLL is unloaded.

● wpclsIncUsage WPObject class method (Warp only)

Increments the usage count of a class by one.

SYNTAX

VOID wpclsIncUsage ()

PARAMETERS

none

RETURNS

none

HOWTO CALL

Call this method on a class object pointer to increment its usage count, thus preventing the system from unloading the class.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpclsDecUsage -69

NOTES

The system uses the class usage count to determine whether the class should be unloaded. The usage count is automatically incremented by the system when an instance is created and decremented when an instance goes dormant. Once the count goes back to zero, the class DLL is unloaded.

● wpclsNew WPObj class method

Creates a new instance of a class.

SYNTAX

WPObj * wpclsNew (**PSZ** *pszTitle*, **PSZ** *pszSetupEnv*, **WPFolder** * *Folder*,
BOOL *fLock*)

PARAMETERS

pszTitle - input

The title of the new object.

pszSetupEnv - input

The initial settings for the new object in the form of a setup string. A setup string is a list of keyname value pairs separated by semicolons which is parsed by the object for initialization. Specify NULL to use the object's defaults. See Appendix A for the list of available keyname value pairs for the predefined Workplace Shell classes. For more information on setup strings, see wpSetup (pg. 63).

Folder - input

The object pointer of the folder in which to place the new object. Some methods for obtaining Folder object pointers are wpclsQueryFolder, wpclsQueryObject, and wpclsQueryObjectFromPath.

fLock - input

TRUE—Increment the lock count on the new object returned.

FALSE—Do not increment the lock count on the new object.

RETURNS

(nonzero value)—The object pointer for the new object.

NULL—An error occurred.

HOW TO CALL

Call this method on any class that supports creating instances. (WPObj and the base classes cannot have instances) This method must be used instead of somNew because Workplace objects require special processing when they are created.

HOW TO OVERRIDE

This method can be overridden for notification when new objects are created, but its functionality cannot be completely replaced so be sure to first call the parent method.

OTHER INFO

Include file: wpobject.h

SEE ALSO

WinCreateObject -9, wpclsQueryFolder -74, wpclsQueryObject -75, wpclsQueryObjectFromPath -76, wpCreateFromTemplate -48, wpDelete -51, wpFree -53, wpSetup -63, wpUnlockObject -65

RESTRICTIONS/WARNINGS

- Pass TRUE for the *fLock* parameter if you are going to access the object pointer after calling wpclsNew. When the pointer is no longer needed, call wpUnlockObject.
- wpclsNew is called by the system whenever a new object is created by any method except wpCreateFromTemplate.

-
- **wpclsObjectFromHandle** **WPObj class**
 method (Warp only)
-

Returns the SOM pointer for an object, given its handle.

SYNTAX

WPObj * wpclsObjectFromHandle (**HOBjECT** *hObject*)

PARAMETERS

hObject - input

The object handle of any Workplace Shell object.

RETURNS

(nonzero value)—The SOM object pointer.

NULL—An error occurred or the object cannot be found.

HOW TO CALL

Call this method on the WPObj class object (`_WPObj`) to get the object pointer for any object. If the object is not awake, Workplace will awaken it and then return the pointer.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`WinQueryObject` -17, `wpclsQueryFolder` -74, `wpclsQueryObject` -75, `wpclsQueryObjectFromPath` -76, `wpUnlockObject` -65

RESTRICTIONS/WARNINGS

- `wpclsObjectFromHandle` will increment the lock count for the returned object pointer. When the pointer is no longer needed, call `wpUnlockObject`.
- If the handle of the object is not known but the object ID is, call `WinQueryObject` on the object ID to obtain the object handle.

●	<code>wpclsQueryAwakeObject</code>	<code>WPFileSystem</code> class method (Warp only)
---	---	--

Tests if a file system object is awake.

SYNTAX

`WPObj * wpclsQueryAwakeObject (PSZ pszInputPath)`

PARAMETERS

pszInputPath - input

The fully qualified path of a file system object.

RETURNS

(nonzero value)—The pointer to the specified object.
 NULL—The object is not awake or was not found.

HOW TO CALL

Call this method on the WPFileSystem class object (_WPFileSystem) to determine if a file system object is awake.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wpfsys.h

SEE ALSO

wpclsQueryFolder -74, wpclsQueryObject -75,
 wpclsQueryObjectFromPath -76

RESTRICTIONS/WARNINGS

- Although the pointer returned from this method is of the type WPObject, this method only returns WPFileSystem objects.

● **wpclsQueryFolder WPObject class method**

Returns the SOM object pointer for a folder, given its location.

SYNTAX

WPFolder * wpclsQueryFolder (**PSZ** *pszLocation*, **BOOL** *fLock*)

PARAMETERS

pszLocation - input

This can be either the object ID of the folder (if it exists and is known) or the fully qualified path of the directory represented by the folder. Refer to Appendix D for a list of default Workplace folder object IDs.

fLock - input

TRUE—Increment the lock count on the object returned.

FALSE—Do not increment the lock count on the object.

RETURNS

(nonzero value)—The object pointer for the folder.

NULL—An error occurred or the object is not a folder.

HOW TO CALL

Call this method on the WPObjct class object (`_WPObjct`) to get the object pointer for a folder. If the folder is not awake, Workplace will awaken it and then return the pointer.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpclsQueryObject` -75, `wpclsQueryObjectFromPath` -76,
`wpUnlockObject` -65

RESTRICTIONS/WARNINGS

- Pass TRUE for the *fLock* parameter if you are going to access the object pointer after calling `wpclsQueryFolder`. When the pointer is no longer needed, call `wpUnlockObject`.

● wpclsQueryObject WPObjct class method

Returns the SOM pointer for an object, given its handle.

SYNTAX

WPObjct * `wpclsQueryObject` (**HOBJECT** *hObject*)

PARAMETERS

hObject - input

The object handle of any Workplace Shell object.

RETURNS

(nonzero value)—The SOM object pointer.

NULL—An error occurred or the object cannot be found.

HOW TO CALL

Call this method on the WPObj class object (`_WPObj`) to get the object pointer for any object. If the object is not awake, Workplace will awaken it and then return the pointer.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`WinQueryObject` -17, `wpclsObjectFromHandle` -72,
`wpclsQueryFolder` -74, `wpclsQueryObjectFromPath` -76,
`wpUnlockObject` -65

RESTRICTIONS/WARNINGS

- `wpclsQueryObject` will increment the lock count for the returned object pointer. When the pointer is no longer needed, call `wpUnlockObject`.
- If the handle of the object is not known but the object ID is, call `WinQueryObject` on the object ID to obtain the object handle.

●	<code>wpclsQueryObjectFromPath</code>	<code>WPFileSystem</code> class method
---	--	---

Returns the SOM object pointer for a file system object, given its location.

SYNTAX

`WPFileSystem * wpclsQueryObjectFromPath (PSZ pszFQPath)`

PARAMETERS

pszFQPath - input

This can be either the object ID of the file system object (if it exists and is known) or the fully qualified path of the file or directory represented by the object. See Appendix D for a list of predefined Workplace object IDs.

RETURNS

(nonzero value)—The requested object pointer.
NULL—An error occurred.

HOW TO CALL

Call this method on the WPFileSystem class object (`_WPFileSystem`) to get the object pointer for a file or folder. If the object is not awake, Workplace will awaken it and then return the pointer.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpfsys.h`

SEE ALSO

`WinQueryObject` -17, `wpclsObjectFromHandle` -72,
`wpclsQueryAwakeObject` -73, `wpclsQueryFolder` -74,
`wpclsQueryObject` -75, `wpUnlockObject` -65

RESTRICTIONS/WARNINGS

- This method will increment the lock count of the returned object pointer. When the pointer is no longer needed, call `wpUnlockObject`.
- There is no equivalent method for abstract objects. To get the object pointer of an abstract given an object ID, call `WinQueryObject` to get the `HOBJECT` and pass that to `wpclsQueryObject` or `wpclsObjectFromHandle` (Warp).

4

Instance and Class Data

Object class definitions can specify both instance and class data. The instance data is allocated for each object to allow them to separately maintain individual information. Only one class object exists for each loaded class; therefore, only one copy of class data resides in memory. Class data and instance data can be obtained by calling the `M_<classname>GetData` macro on the class object or the `<classname>GetData` macro on an object, respectively.

Initializing and Uninitializing Data

When a class is loaded, the system calls `wpclsInitData` to allow the class to allocate and initialize any required resources for its data (such as memory or module handles). As each instance is awakened, `wpInitData` is called to allow the object to allocate and initialize its instance data. `wpclsInitData` and `wpInitData` should be overridden by the subclass to allocate and initialize class and instance data defined by the subclass.

When a class is unloaded, the system calls `wpclsUnInitData` to allow the class to deallocate its resources. `wpUnInitData` is called on each instance as it is made dormant to allow the object to deallocate its instance data. If resources are allocated in a `wpclsInitData` override, `wpclsUnInitData` must also be overridden to free them. Furthermore, any resources allocated in a `wpInitData` override must be freed from within `wpUnInitData`.

Saving Data

When the instance data of an object has changed and must be saved, call the `wpSaveDeferred` method to mark the object as “dirty.” The Workplace Shell has a separate thread for saving dirty objects which, after approximately five seconds, will call `wpSaveImmediate` on each one. `wpSaveImmediate` can also be called directly on an object to cause it to save its state immediately instead of asynchronously, but this can affect the performance of the system. The base classes `WPFileSystem` and `WPAbstract` provide persistent storage for instance data and process `wpSaveImmediate` by calling `wpSaveState` and `wpRestoreState` on the dirty object. These methods should be overridden to save and restore instance data, respectively.

Workplace does not provide methods for saving class data. Classes must provide their own storage mechanisms for saving and restoring this data. Class data should be restored in the override of `wpclsInitData` and should be saved whenever the data is changed.

Restrictions and Warnings

- Class data should not be saved in the `wpclsUnInitData` method override because this method is not called when the user shuts down the system.

Instance Methods

wpAllocMem allocates a block of memory of a specified size (pg. 80).

wpFreeMem frees memory allocated with **wpAllocMem** (pg. 82).

wpInitData initializes an object's data when created (pg. 83).

wpRestoreData restores instance data stored as a block of binary data (pg. 84).

wpRestoreLong restores instance data stored as a long (pg. 87).

wpRestoreState notifies the object to restore its instance data (pg. 89).

wpRestoreString restores instance data stored as a string (pg. 90).

wpSaveData saves instance data as a block of binary data (pg. 92).

wpSaveDeferred marks an object to save its state asynchronously (pg. 94).

wpSaveImmediate causes an object to save its state synchronously (pg. 95).

wpSaveLong saves instance data as a long (pg. 96).

wpSaveState notifies an object to save its instance data (pg. 97).

wpSaveString saves instance data as a string (pg. 98).

wpUnInitData uninitializes an object's data (pg. 100).

● **wpAllocMem** **WPObject instance method**

Allocates a block of memory of a specified size.

SYNTAX

PBYTE wpAllocMem (**ULONG** *cbBytes*, **PULONG** *prc*)

PARAMETERS

cbBytes - input

The size, in bytes, of the requested block of memory.

prc - output

A pointer to the error returned from **DosAllocMem**. Specify **NULL** and Workplace will display the error to the user instead.

RETURNS

(nonzero value)—The requested block of memory.

NULL—An error occurred.

HOW TO CALL

Call this method on any object to allocate memory. When the memory is no longer needed, call `wpFreeMem`. The system will call this method to allocate memory for objects.

HOW TO OVERRIDE

Override this method to replace the memory allocation scheme for your objects. Do not call the parent method. If this method is overridden, `wpFreeMem` must also be overridden so the memory will be properly freed.

OTHER INFO

Include file: `wobject.h`

SEE ALSO

`wpAddToObjUseList` -145, `wpFreeMem` -82, `wpUnInitData` -100

NOTES

When this method is called, the system puts a `USAGE_MEMORY` item in the in-use list for the object. See Chapter 6 for more information about in-use lists. If the object is deleted or goes dormant, `wpUnInitData` is called and the `WObject` implementation for `wpUnInitData` will look through the in-use list for `USAGE_MEMORY` items and call `wpFreeMem` on each one that is found.

If this method is overridden, the replacing implementation should also put a `USAGE_MEMORY` item in the in-use list so that the system can detect memory that has not been freed when `wpUnInitData` is called.

RESTRICTIONS/WARNINGS

- Since `WObject` frees any memory allocated with `wpAllocMem` in `wpUnInitData`, a class that overrides `wpUnInitData` should either not free any of this memory still allocated, or free it with `wpFreeMem` *before* calling the parent method. Otherwise, the memory will be freed twice.

SAMPLE CODE

The following sample is a method called `SetComment` defined by the class `MyClass`. The purpose of this method is to set an instance variable called `pszComment` using `wpAllocMem` to allocate the memory.

```

SOM_Scope BOOL  SOMLINK mcls_SetComment(MyClass *somSelf,
                                         PSZ pszNewString)
{
    BOOL      bReturn = TRUE;
    MyClassData *somThis = MyClassGetData(somSelf);

    /* if _pszComment is already set to something, free it */
    if (_pszComment)
        _wpFreeMem(somSelf, (PBYTE) _pszComment);

    /* if pszNewString is not NULL, allocate memory for the string and
       copy the new string. Otherwise just set pszComment to NULL.
       _pszComment should be freed in the wpUnInitData override if it is
       non-NULL.
       */
    if (pszNewString)
    {
        _pszComment = _wpAllocMem(somSelf, strlen(pszNewString) + 1,
                                   NULL);

        if (_pszComment)
            strcpy(_pszComment, pszNewString);
        else
            /* error allocating memory */
            bReturn = FALSE;
    }
    else
        _pszComment = NULL;
    return(bReturn);
}

```

● **wpFreeMem** **WPObject** instance method

Frees memory allocated with wpAllocMem.

SYNTAX

BOOL wpFreeMem (**PBYTE** *pByte*)

PARAMETERS

pByte - input

A pointer to the block of memory to be freed. It must have been allocated using `wpAllocMem`.

RETURNS

TRUE—The memory was freed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method on an object to free memory that was allocated with `wpAllocMem`. The system also calls this method to free an object's memory.

HOW TO OVERRIDE

Override this method to replace the memory allocation scheme for your objects if the `wpAllocMem` method was also overridden. Do not call the parent method. The `wpAllocMem` override should have added `USAGE_MEMORY` items to the in-use list. Remove the `USAGE_MEMORY` item that corresponds to the memory block that is freed.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpAllocMem` -80, `wpDeleteFromObjUseList` -148, `wpUnInitData` -100

RESTRICTIONS/WARNINGS

- Since `WPObj` frees any memory allocated with `wpAllocMem` in `wpUnInitData`, a class that overrides `wpUnInitData` should either not free any of this memory still allocated, or free it with `wpFreeMem` *before* calling the parent method. Otherwise, the memory will be freed twice.

● `wpInitData` **WPObj** instance method

Initializes an object's data when created.

SYNTAX

VOID `wpInitData` ()

PARAMETERS

none

RETURNS

none

HOW TO CALL

This method is called only by the system.

HOW TO OVERRIDE

Override this method to initialize an object to a known state. The parent method should be called first, which will initialize all of the instance data to zero. `wpUnInitData` should be overridden to free any memory that was allocated in this override.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpAllocMem` -80, `wpFreeMem` -82, `wpSetup` -63, `wpSetupOnce` -64, `wpUnInitData` -100

NOTES

The system calls this method each time an object is created or awakened. If there is setup processing for an object that needs to be done only once, use the `wpSetupOnce` (Warp) method.

RESTRICTIONS/WARNINGS

- `wpInitData` is called before an object is initialized. Since many Workplace methods do not work properly unless an object is initialized, calling predefined Workplace methods from this override is not advised.

● **`wpRestoreData` `WPObj` instance method**

Restores instance data stored as a block of binary data.

SYNTAX

BOOL `wpRestoreData` (**PSZ** *pszClass*, **ULONG** *ulKey*, **PBYTE** *pValue*,
PULONG *pcbValue*)

PARAMETERS

pszClass - input

The unique string that represents this class. It is recommended that the class name be used because it will be unique in the system. All calls to save and restore methods from within a class implementation must use the same *pszClass* value. This parameter is case-sensitive.

ulKey - input

The application-defined number that represents the desired instance data to restore.

pValue - input/output

The buffer to hold the requested data. If NULL is specified, the required size will be returned in *pcbValue*.

pcbValue - input/output

The address of the size of the *pValue* buffer. If NULL is specified for *pValue*, the required size is returned in this parameter.

RETURNS

TRUE—The data for the specified *ulKey* was found and the call was successful.

FALSE—An error occurred or the specified *ulKey* was not found.

HOW TO CALL

This method can only be called from an override of `wpRestoreState` to restore data that was saved with `wpSaveData`.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wobject.h`

SEE ALSO

`wpRestoreLong` -87, `wpRestoreState` -89, `wpRestoreString` -90, `wpSaveData` -92, `wpSaveDeferred` -94, `wpSaveImmediate` -95, `wpSaveLong` -96, `wpSaveState` -97, `wpSaveString` -98

NOTES

The values used for *ulKey* are defined by the class and must remain consistent. Each key is an index into the data created by the storage class. If an instance variable is saved using a specific key, that variable must be restored with the same value.

RESTRICTIONS/WARNINGS

- If the format of the data should change or if the instance data is removed from the class definition, the key that was used to represent this data cannot be reused because it would break existing instances of the class on the user's machine. Applications that already have production versions of classes that customers are using must always define new keys that have never been used before.

SAMPLE CODE

This sample is an override of the `wpRestoreState` method to demonstrate restoring instance data using `wpRestoreData` for a class called `MyClass`.

```
#define IDKEY_DATA 1 // the next keys would be 2,3,4 and so on
CHAR szClassName[] = "MyClass";

SOM_Scope BOOL  SOMLINK mcls_wpRestoreState(MyClass *somSelf,
                                             ULONG ulReserved)
{
    ULONG    ulSize;
    MyClassData *somThis = MyClassGetData(somSelf);
    /* restore the PVOID instance data called pData */
    if ( _wpRestoreData(somSelf, szClassName, IDKEY_DATA, NULL,
                        &ulSize))
    {
        _pData = _wpAllocMem(somSelf, ulSize, NULL);
        if ( _pData)
        {
            _wpRestoreData(somSelf, szClassName, IDKEY_DATA, _pData,
                          &ulSize);
            /* save the size in the ULONG instance data called _ulDataSize */
            _ulDataSize = ulSize;
        }
    }
    return (parent_wpRestoreState(somSelf,ulReserved));
}
```

wpRestoreLong WPObject instance method

Restores instance data stored as a long.

SYNTAX

BOOL wpRestoreLong (**PSZ** *pszClass*, **ULONG** *ulKey*, **PULONG** *pulValue*)

PARAMETERS

pszClass - input

The unique string that represents this class. It is recommended that the class name be used because it will be unique in the system. All calls to save and restore methods from within a class implementation must use the same *pszClass* value. This parameter is case-sensitive.

ulKey - input

The application-defined number that represents the desired instance data to restore.

pulValue - output

The address of the returned long value.

RETURNS

TRUE—The value for the specified *ulKey* was found and the call was successful.

FALSE—An error occurred or the specified *ulKey* was not found.

HOW TO CALL

This method can only be called from an override of wpRestoreState to restore data that was saved with wpSaveLong.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpRestoreData -84, wpRestoreState -89, wpRestoreString -90,
wpSaveData -92, wpSaveDeferred -94, wpSaveImmediate -95,
wpSaveLong -96, wpSaveState -97, wpSaveString -98

NOTES

The values used for *ulKey* are defined by the class and must remain consistent. Each key is an index into the data created by the storage class. If an instance variable is saved using a specific key, that variable is restored with the same value.

RESTRICTIONS/WARNINGS

- If the format of the data should change or if the instance data is removed from the class definition, the key that was used to represent this data cannot be reused because it would break existing instances of the class on the user's machine. Applications that already have production versions of classes that customers are using must always define new keys that have never been used before.

SAMPLE CODE

This sample is an override of the `wpRestoreState` method to demonstrate restoring instance data using `wpRestoreLong` for a class called `MyClass`.

```
#define IDKEY_FLAG 2 // the next keys would be 3,4 and so on
CHAR szClassName[] = "MyClass";

SOM_Scope BOOL  SOMLINK mcls_wpRestoreState(MyClass *somSelf,
                                             ULONG ulReserved)
{
    ULONG    ulValue;
    MyClassData *somThis = MyClassGetData(somSelf);

    /* restore the ULONG instance data called ulFlag. */

    if ( _wpRestoreLong(somSelf, szClassName, IDKEY_FLAG,
                        &ulValue))
    {
        /* do not set _ulFlag unless the data was previously saved */

        _ulFlag = ulValue;
    }
    return (parent_wpRestoreState(somSelf,ulReserved));
}
```

● wpRestoreState WPyObject instance method

Notifies an object to restore its instance data.

SYNTAX

BOOL wpRestoreState (**ULONG** *ulReserved*)

PARAMETERS

ulReserved - reserved

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can only be called by the system when an object is awakened.

HOW TO OVERRIDE

Override this method to restore data that was saved in wpSaveState. Use wpRestoreData, wpRestoreLong, and wpRestoreString. You must call the parent method.

OTHER INFO

Include file: wpyobject.h

SEE ALSO

wpRestoreData -84, wpRestoreLong -87, wpRestoreString -90, wpSaveData -92, wpSaveDeferred -94, wpSaveImmediate -95, wpSaveLong -96, wpSaveState -97, wpSaveString -98

NOTES

wpRestoreState should only be overridden if the wpSaveState method was also overridden to save instance data.

SAMPLE CODE

See the samples for wpRestoreData, wpRestoreLong, and wpRestoreString.

● **wpRestoreString** **WPObj**ect instance method

Restores instance data stored as a string.

SYNTAX

BOOL wpRestoreString (**PSZ** *pszClass*, **ULONG** *ulKey*, **PSZ** *pszValue*,
PULONG *pcbValue*)

PARAMETERS

pszClass - input

The unique string that represents this class. It is recommended that the class name be used because it will be unique in the system. All calls to save and restore methods from within a class implementation must use the same *pszClass* value. This parameter is case-sensitive.

ulKey - input

The application-defined number that represents the desired instance data to restore.

pszValue - input/output

The buffer to hold the requested string. If NULL is specified, the required size will be returned in *pcbValue*.

pcbValue - input/output

The address of the size of the *pszValue* buffer. If NULL is specified for *pszValue*, the required size is returned in this parameter.

RETURNS

TRUE—The value for the specified *ulKey* was found and the call was successful.

FALSE—An error occurred or the specified *ulKey* was not found.

HOW TO CALL

This method can only be called from an override of wpRestoreState to restore data that was saved with wpSaveString.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpRestoreData -84, wpRestoreLong -87, wpRestoreState -89, wpSaveData -92, wpSaveDeferred -94, wpSaveImmediate -95, wpSaveLong -96, wpSaveState -97, wpSaveString -98

NOTES

The values used for *ulKey* are defined by the class and must remain consistent. Each key is an index into the data created by the storage class. If an instance variable is saved using a specific key, that variable must be restored with the same value.

RESTRICTIONS/WARNINGS

- If the format of the data should change or if the instance data is removed from the class definition, the key that was used to represent this data cannot be reused because it would break existing instances of the class on the user's machine. Applications that already have production versions of classes that customers are using must always define new keys that have never been used before.

SAMPLE CODE

This sample is an override of the wpRestoreState method to demonstrate restoring instance data using wpRestoreString for a class called MyClass.

```
#define IDKEY_COMMENT 3 // the next keys would be 4,5,6 and so on
CHAR szClassName[] = "MyClass";

SOM_Scope BOOL  SOMLINK mcls_wpRestoreState(MyClass *somSelf,
                                             ULONG ulReserved)
{
    ULONG    ulSize;
    MyClassData *somThis = MyClassGetData(somSelf);

    /* restore the PSZ instance data called pszComment */

    if ( _wpRestoreString(somSelf, szClassName, IDKEY_COMMENT, NULL,
                          &ulSize))
    {
        _pszComment = (PVOID) _wpAllocMem(somSelf, ulSize, NULL);

        if ( _pszComment)
```

```

        _wpRestoreString(somSelf, szClassName, IDKEY_COMMENT,
                        _pszComment, &ulSize);
    }
    return (parent_wpRestoreState(somSelf, ulReserved));
}

```

● **wpSaveData** **WPObj**ect instance method

Saves instance data as a block of binary data.

SYNTAX

BOOL wpSaveData (**PSZ** *pszClass*, **ULONG** *ulKey*, **PBYTE** *pValue*,
ULONG *cbValue*)

PARAMETERS

pszClass - input

The unique string that represents this class. It is recommended that the class name be used because it will be unique in the system. All calls to save and restore methods from within a class implementation must use the same *pszClass* value. This parameter is case-sensitive.

ulKey - input

The application-defined number that represents the desired instance data to save.

pValue - input

The buffer that holds the data to save.

cbValue - input

The size of the *pValue* buffer.

RETURNS

TRUE—The data was saved successfully.

FALSE—An error occurred.

HOW TO CALL

This method can only be called from an override of wpSaveState.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpRestoreData -84, wpRestoreLong -87, wpRestoreState -89,
wpRestoreString -90, wpSaveDeferred -94, wpSaveImmediate -95,
wpSaveLong -96, wpSaveState -97, wpSaveString -98

NOTES

The values used for *ulKey* are defined by the class and must remain consistent. Each key is an index into the data created by the storage class. If an instance variable is saved using a specific key, that variable must be restored with the same value.

RESTRICTIONS/WARNINGS

- When instance data is saved from the wpSaveState method, the entire block of data is re-created and overwrites the old one. Therefore, if some data was saved on a previous call to wpSaveState but should be removed because it is zero or no longer valid, simply refrain from saving the data in the wpSaveState override (see sample below).
- If the format of the data should change or if the instance data is removed from the class definition, the key that was used to represent this data cannot be reused because it would break existing instances of the class on the user's machine. Applications that already have production versions of classes that customers are using must always define new keys that have never been used before.

SAMPLE CODE

This sample is an override of the wpSaveState method to demonstrate saving instance data using wpSaveData for a class called MyClass. The data being saved is called pData, and the instance data ulDataSize is used to keep track of the size of pData. (See the wpRestoreData sample.)

```
#define IDKEY_DATA 1 // the next keys would be 2,3,4 and so on
CHAR szClassName[>] = "MyClass";
SOM_Scope BOOL SOMLINK mcls_wpSaveState(MyClass *somSelf)
{
    MyClassData *somThis = MyClassGetData(somSelf);
    /* save the data only if it is non zero. If the data has been
       saved at some point but is now zero, just avoiding calling
       wpSaveData this time will erase the old data.
    */
}
```

```
if (_pData && _ulDataSize)
{
    _wpSaveData(somSelf, szClassName, IDKEY_DATA, _pData,
                _ulDataSize);
}
return (parent_wpSaveState(somSelf));
}
```

● **wpSaveDeferred** **WPObject instance method**

Marks an object to save its state asynchronously.

SYNTAX

BOOL wpSaveDeferred ()

PARAMETERS

none

RETURNS

TRUE—The call was successful.

FALSE—An error occurred.

HOW TO CALL

Call this method on any object to notify Workplace to save the object asynchronously. After approximately five seconds, a separate thread will call wpSaveImmediate on the object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpRestoreState -89, wpSaveImmediate -95, wpSaveState -97

NOTES

Any time persistent data of an object has been changed, the wpSaveDeferred method must be called or the data may not be saved if the object goes dormant or the machine is rebooted. If the object

must be saved immediately, the `wpSaveImmediate` method can be called directly but could affect system performance.

The system will call the `wpSaveImmediate` method if the object is still waiting to be saved and it goes dormant or the system is shut down.

● **`wpSaveImmediate`** **WObject instance method**

Marks an object to save its state synchronously.

SYNTAX

BOOL `wpSaveImmediate` ()

PARAMETERS

none

RETURNS

TRUE—The call was successful.

FALSE—An error occurred.

HOW TO CALL

Call this method on any object to notify Workplace to save the object immediately. This method should only be called directly if very critical data has been altered that must be saved right away. Otherwise, use `wpSaveDeferred`.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wobject.h`

SEE ALSO

`wpRestoreState` -89, `wpSaveDeferred` -94, `wpSaveState` -97

NOTES

Any time persistent data of an object has been changed, the `wpSaveDeferred` method must be called or the data will not be saved if the object goes dormant or the machine is rebooted. If the object must be saved immediately, the `wpSaveImmediate` method can be called directly but could affect system performance.

● **wpSaveLong** **WPObject instance method**

Saves instance data stored as a long.

SYNTAX

BOOL wpSaveLong (**PSZ** *pszClass*, **ULONG** *ulKey*, **ULONG** *ulValue*)

PARAMETERS

pszClass - input

The unique string that represents this class. It is recommended that the class name be used because it will be unique in the system. All calls to save and restore methods from within a class implementation must use the same *pszClass* value. This parameter is case-sensitive.

ulKey - input

The application-defined number that represents the desired instance data to save.

ulValue - output

The value to save.

RETURNS

TRUE—The data was saved successfully.

FALSE—An error occurred.

HOW TO CALL

This method can only be called from an override of wpSaveState.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpRestoreData -84, wpRestoreLong -87, wpRestoreState -89,
wpRestoreString -90, wpSaveData -92, wpSaveDeferred -94,
wpSaveImmediate -95, wpSaveState -97, wpSaveString -98

NOTES

The values used for *ulKey* are defined by the class and must remain consistent. Each key is an index into the data created by the storage

class. If an instance variable is saved using a specific key, that variable is restored with the same value.

RESTRICTIONS/WARNINGS

- If the format of the data should change or if the instance data is removed from the class definition, the key that was used to represent this data cannot be reused because it would break existing instances of the class on the user's machine. Applications that already have production versions of classes that customers are using must always define new keys that have never been used before.

SAMPLE CODE

This sample is an override of the `wpSaveState` method to demonstrate saving instance data using `wpSaveLong` for a class called `MyClass`.

```
#define IDKEY_FLAG 2 // the next keys would be 3,4 and so on
CHAR szClassName<[>] = "MyClass";

SOM_Scope BOOL SOMLINK mcls_wpSaveState(MyClass *somSelf)
{
    MyClassData *somThis = MyClassGetData(somSelf);

    _wpSaveLong(somSelf, szClassName, IDKEY_FLAG, _ulFlag);

    return (parent_wpSaveState(somSelf));
}
```

● **wpSaveState** **WPObject** instance method

Notifies an object to save its instance data.

SYNTAX

BOOL wpSaveState ()

PARAMETERS

none

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can only be called by the system when an object's state is being saved.

HOW TO OVERRIDE

Override this method to save instance data of the object by calling `wpSaveData`, `wpSaveLong`, and `wpSaveString`. You must call the parent method.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpRestoreData` -84, `wpRestoreLong` -87, `wpRestoreState` -89,
`wpRestoreString` -90, `wpSaveData` -92, `wpSaveDeferred` -94,
`wpSaveImmediate` -95, `wpSaveLong` -96, `wpSaveString` -98

NOTES

`wpRestoreState` should be overridden to restore any data saved from the `wpSaveState` override.

SAMPLE CODE

See the samples for `wpSaveData`, `wpSaveLong`, and `wpSaveString`.

● **`wpSaveString`** **WPObj** instance method

Saves instance data as a string.

SYNTAX

BOOL `wpSaveString` (**PSZ** *pszClass*, **ULONG** *ulKey*, **PSZ** *pszValue*)

PARAMETERS

pszClass - input

The unique string that represents this class. It is recommended that the class name be used because it will be unique in the system. All calls to save and restore methods from within a class implementation must use the same *pszClass* value. This parameter is case-sensitive.

ulKey - input

The application-defined number that represents the desired instance data to save.

pszValue - input

A pointer to the string to save. This pointer cannot be NULL.

RETURNS

TRUE—The string was saved successfully.

FALSE—An error occurred.

HOW TO CALL

This method can only be called from an override of `wpSaveState`.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wobject.h`

SEE ALSO

`wpRestoreData` -84, `wpRestoreLong` -87, `wpRestoreState` -89,
`wpRestoreString` -90, `wpSaveData` -92, `wpSaveDeferred` -94,
`wpSaveImmediate` -95, `wpSaveLong` -96, `wpSaveState` -97

NOTES

The values used for *ulKey* are defined by the class and must remain consistent. Each key is an index into the data created by the storage class. If an instance variable is saved using a specific key, that variable must be restored with the same value.

RESTRICTIONS/WARNINGS

- When instance data is saved from the `wpSaveState` method, the entire block of data is re-created and overwrites the old one. Therefore, if some data was saved on a previous call to `wpSaveState` but should be removed because it is zero or no longer valid, simply refrain from saving the data in the `wpSaveState` override (see sample below).
- If the format of the data should change or if the instance data is removed from the class definition, the key that was used to represent this data cannot be reused because it would break existing instances of the class on the user's machine. Applications that already have production versions of classes that customers are using must always define new keys that have never been used before.

SAMPLE CODE

This sample is an override of the wpSaveState method to demonstrate saving instance data using wpSaveString for a class called MyClass.

```
#define IDKEY_COMMENT 3 // the next keys would be 4,5,6 and so on
CHAR szClassName[] = "MyClass";

SOM_Scope BOOL  SOMLINK mcls_wpSaveState(MyClass *somSelf)
{
    MyClassData *somThis = MyClassGetData(somSelf);
    /* only save the data if pszComment is nonzero. */
    if (_pszComment)
        _wpSaveString(somSelf, szClassName, IDKEY_COMMENT, _pszComment);

    return (parent_wpSaveState(somSelf));
}
```

● wpUnInitData WPObject instance method

Uninitializes an object's data.

SYNTAX

VOID wpUnInitData ()

PARAMETERS

none

RETURNS

none

HOW TO CALL

This method is only called by the system.

HOW TO OVERRIDE

Override this method to free any memory or other resources allocated for an object. Call the parent method last.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpAllocMem -80, wpFreeMem -82, wpInitData -83, wpSetup -63

NOTES

The system calls this method each time an object is deleted or made dormant.

Class Methods

wpclsInitData initializes the class object's data (pg. 101).

wpclsUnInitData uninitializes the class object's data (pg. 102).

● **wpclsInitData** **WPyObject class method**

Initializes a class object's data.

SYNTAX

VOID wpclsInitData ()

PARAMETERS

none

RETURNS

none

HOW TO CALL

This method is only called by the system.

HOW TO OVERRIDE

Override this method to initialize any class variables or to allocate any memory or other resources for the class. Call the parent method first. Override wpclsUnInitData to free these resources.

OTHER INFO

Include file: wpyobject.h

SEE ALSO

wpclsUnInitData -102

NOTES

A Workplace class is loaded when it is registered or when its first instance is awakened. The system calls this method when the class object is created.

RESTRICTIONS/WARNINGS

- When this method is invoked, no instances of the class yet exist. Therefore, the instance method, wpAllocMem, cannot be used to allocate memory from this override. Use DosAllocMem or another method of memory allocation.

● wpclsUnInitData WPObject class method

Uninitializes a class object's data.

SYNTAX

VOID wpclsUnInitData ()

PARAMETERS

none

RETURNS

none

HOW TO CALL

This method is only called by the system.

HOW TO OVERRIDE

Override this method to free any resources allocated for this class. Call the parent method last.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpclsInitData -102

NOTES

When a class is no longer needed in the system because the last instance has gone dormant, the system calls this method just before the class DLL is unloaded.

RESTRICTIONS/WARNINGS

- Do not load or link with any DLLs that register an exit list. When SOM tries to free the class DLL, it will fail because the Workplace process is still active. This will prevent any objects of this class from waking up after the class has been unloaded.

5

General Object Settings

Some instance data can be set for all Workplace Shell objects. Examples include the object's title, icon, details, or style. The `WPObject` class defines these attributes and provides methods for setting and querying the data. Most of these settings have class defaults as well as individual instance values. For example, `wpclsQueryTitle` will return the default title for all members of the class but `wpQueryTitle` returns the title of a specific instance. By default, objects will use the class defaults.

Object Icons

There are multiple methods for setting and querying the icon that represents an object, and it is sometimes unclear when to use which methods. Furthermore, the mechanism for setting default class data differs from the mechanism for setting the instance data.

The class default icon should be specified by overriding `wpclsQueryIconData` and returning the desired icon. `wpclsQueryIcon` is a method that was made public before `wpclsQueryIconData` was written and has now become unnecessary. The icon for an individual object can be queried with either `wpQueryIcon` or `wpQueryIconData`. The icon can be altered by calling `wpSetIcon` to temporarily change the icon for the object or `wpSetIconData` followed by `wpSaveDeferred` for a persistent change. In Warp, `wpclsSetIconData` and `wpclsSetIcon` can be used to temporarily change the default icon of a class.

Device Objects

The device object classes `WPMouse`, `WPKeyboard`, and `WPSystem` define setting strings that can be set and queried using `wpclsSetSetting` and `wpclsQuerySetting` as defined by the `WPAbstract` class. These methods enable applications to modify system default information without having to directly access the user ini file.

Restrictions and Warnings

- Once a persistent object is created, even if it is only using class default settings, all data is saved to the object itself. This means that if you then change the default settings for your class, it will not necessarily affect instances that have already been created.

Instance Methods

`wpCnrRefreshDetails` (Warp) refreshes the details data of an object (pg. 106).

`wpCnrSetEmphasis` modifies the container record attributes for an object (pg. 107).

`wpModifyStyle` changes the style of an object (pg. 108).

`wpQueryConfirmations` returns the confirmations set for an object (pg. 111).

wpQueryCoreRecord (Warp) returns a pointer to the container record for an object (pg. 112).

wpQueryDetailsData returns the details data of an object (pg. 113).

wpQueryIcon returns the handle for the icon representing an object (pg. 114).

wpQueryIconData returns the data for the icon representing an object (pg. 115).

wpQueryObjectID (Warp) returns the ID string of an object (pg. 117).

wpQueryStyle returns the style of an object (pg. 118).

wpQueryTitle returns the title of an object (pg. 118).

wpSetIcon sets the handle for the icon representing an object (pg. 119).

wpSetIconData sets the data for the icon representing an object (pg. 120).

wpSetStyle sets the style of an object (pg. 121).

wpSetTitle sets the title of an object (pg. 122).

●	wpCnrRefreshDetails	WPObject instance method (Warp only)
---	----------------------------	---

Refreshes the details data of an object.

SYNTAX

VOID wpCnrRefreshDetails ()

PARAMETERS

none

RETURNS

none

HOW TO CALL

This method should be called when an object's details data have been modified. This will refresh the data displayed in the details view of the object's folder.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wobject.h`

SEE ALSO

`wpQueryDetailsData` -113

❶ wpCnrSetEmphasis WPObject instance method

Modifies the container record attributes for an object.

SYNTAX

BOOL wpCnrSetEmphasis(**ULONG** *ulEmphasisAttr*, **BOOL** *fTurnOn*)

PARAMETERS

ulEmphasisAttr - input

The record attribute to modify for this object. Choose one from the following list:

Constant		Description
<code>CRA_SELECTED</code>	<code>0x001</code>	Record is selected.
<code>CRA_TARGET</code>	<code>0x002</code>	Record has target emphasis.
<code>CRA_CURSORED</code>	<code>0x004</code>	Cursor is on the record.
<code>CRA_INUSE</code>	<code>0x008</code>	Record has in-use emphasis.
<code>CRA_FILTERED</code>	<code>0x010</code>	Record is invisible.
<code>CRA_DROPONABLE</code>	<code>0x020</code>	Record can accept a drop.
<code>CRA_RECORDREADONLY</code>	<code>0x040</code>	Record is read-only.
<code>CRA_EXPANDED</code>	<code>0x080</code>	Record is expanded (parent records in tree view only).
<code>CRA_COLLAPSED</code>	<code>0x100</code>	Record is collapsed (parent records in tree view only).

fTurnOn - input

Pass **TRUE** to set the flags in *ulEmphasisAttr* and pass **FALSE** to remove them.

RETURNS

TRUE—The record attributes were successfully modified.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time to modify the container record attributes for an object. These attributes are used for all the views that contain this object. The system calls this method to set in-use emphasis on objects and to filter objects excluded by the Include page of its folder's settings notebook.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wobject.h` and `pmstdldlg.h` define:
 `INCL_WINSTDCNR`

● **wpModifyStyle WPObj instance method**

Changes the style of an object.

SYNTAX

BOOL wpModifyStyle (**ULONG** *ulStyleFlags*, **ULONG** *ulStyleMask*)

PARAMETERS

ulStyleFlags - input

The style flags to be modified for this object. OR together multiple style flags to turn on or off more than one flag at once. The following is a list of object style flags:

Constant		Description
OBJSTYLE_NOMOVE	0x02	This object cannot be moved by the user. The Move item will not appear in the pop-up menu.
OBJSTYLE_NOLINK	0x04	A shadow of this object cannot be made by the user. The Create shadow item will not appear in the pop-up menu.

OBJSTYLE_NOCOPY	0x08	This object cannot be copied by the user. The Copy item will not appear in the pop-up menu.
OBJSTYLE_NOTDEFAULTICON	0x10	The icon for this object should be destroyed when the object goes dormant. (Use OBJSTYLE_CUSTOMICON in Warp.)
OBJSTYLE_TEMPLATE	0x20	This object is a template. The default drag operation for a template object is to create a new instance of the same class.
OBJSTYLE_NODELETE	0x40	This object cannot be deleted by the user. The Delete item will not appear in the pop-up menu.
OBJSTYLE_NOPRINT	0x80	This object does not support the print operation. The Print item will not appear in the pop-up menu.
OBJSTYLE_NODRAG	0x100	This object cannot be dragged by the user.
OBJSTYLE_NOTVISIBLE	0x200	This object is not visible. The record for this object is placed in an open folder but is filtered so it is invisible.
OBJSTYLE_NOSETTINGS	0x400	The settings view is not supported by this object.
OBJSTYLE_NORENAME	0x800	This object cannot be renamed by the user.
OBJSTYLE_NODROP	0x1000	This object cannot have other objects dropped on it. (Use OBJSTYLE_NODROPON in Warp.)
OBJSTYLE_NODROPON	0x2000	Use this instead of OBJSTYLE_NODROP (Warp only).
OBJSTYLE_CUSTOMICON	0x4000	Use this instead of OBJSTYLE_NOTDEFAULTICON (Warp only).

ulStyleMask - input

The mask of flags indicating which styles specified in *ulStyleFlags* will be added and which will be removed. For example, if *ulStyleFlags* is `OBJSTYLE_NODELETE | OBJSTYLE_TEMPLATE` and *ulStyleMask* is `OBJSTYLE_NODELETE`, the `OBJSTYLE_NODELETE` flag will be added to the style of this object and the `OBJSTYLE_TEMPLATE` flag will be removed. The valid flags for this parameter are the same as those for *ulStyleFlags*.

RETURNS

TRUE—The call was successful.

FALSE—An error occurred.

HOW TO CALL

Call this method on any object to change its style. This method should be used instead of `wpSetStyle` because it affects only those styles specified in *ulStyleFlags*.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`WinCreateObject` -9, `WinSetObjectData` -22, `wpclsNew` -71, `wpCreateAnother` -47, `wpFilterPopupMenu` -193, `wpQueryStyle` -118, `wpSetStyle` -121, `wpSetup` -63

NOTES

The styles of an object can also be set with a setup string passed to functions or methods that support this parameter. See Appendix A for a list of keyname value pairs for setup strings.

Many of the object styles affect whether or not certain pop-up menu items appear for an object. For example, `OBJSTYLE_NODELETE` prevents the Delete item from appearing. It is better to set the style on the object than to just filter the corresponding item in the `wpFilterPopupMenu` override.

RESTRICTIONS/WARNINGS

- The style `OBJSTYLE_NOSETTINGS` has no effect in OS/2 release 2.1 and earlier. This was fixed in version 2.11.

SAMPLE CODE

The following sample demonstrates how to call `wpModifyStyle`. This method can be called at any time on any Workplace object from either a function or a method override. In this example, the variable `Object` is of the type `SOMAny *` and is assumed to have already been set to a valid object pointer.

```
/* make this object visible and prevent the user from renaming it */
_wpModifyStyle(Object, OBJSTYLE_NOTVISIBLE | OBJSTYLE_NORENAME,
               OBJSTYLE_NORENAME);
```

● wpQueryConfirmations WPObject instance method

Returns the user action confirmations set for an object.

SYNTAX

ULONG wpQueryConfirmations ()

PARAMETERS

none

RETURNS

Constant		Description
CONFIRM_DELETE	0x01	Confirm the deletion of this object.
CONFIRM_DELETEFOLDER	0x02	Confirm folder deletion. This is ignored if this is not a folder.
CONFIRM_RENAME FILESWITHEXT	0x04	Confirm the renaming of this object if it has an extension.
CONFIRM_KEEPPASSOC	0x08	Confirm the renaming of data files with extensions that are associated with an application.
CONFIRM_ACTION	0x10	Confirm copy, move, and create shadow operations.
CONFIRM_PROGRESS	0x20	Show the progress dialog for copy, move, or create shadow operations.

HOW TO CALL

Call this method on any object to query its confirmation flags. Query the delete confirmation flags of an object before deleting it to check if the user wishes to be notified. The default behavior in WPObject for this method is to return the user preferences set on the Confirmations page of the System object.

HOW TO OVERRIDE

Override this method to change the value returned for an object. Call the parent method first to get the user preferences and then modify the return code.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpConfirmDelete -40, wpDelete -51, wpFree -53

●	wpQueryCoreRecord	WPObject instance method (Warp only)
---	--------------------------	---

Returns a pointer to the container record for an object.

SYNTAX

PMINIRECORDCORE wpQueryCoreRecord ()

PARAMETERS

none

RETURNS

A pointer to the container record for the object. Do not free this memory.

HOW TO CALL

Call this method on any object to access its container record directly. Call wpCnrSetEmphasis instead to change any of the record's emphasis attributes.

HOW TO OVERRIDE

This method should not be overridden.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpCnrSetEmphasis -107`

❶ **wpQueryDetailsData** **WPObj** instance method

Returns the details data of an object.

SYNTAX

ULONG wpQueryDetailsData (**PVOID** **ppDetailsData*, **PULONG** *pcp*)

PARAMETERS

ppDetailsData - input

This pointer contains a pointer to a buffer that was allocated by the system for this object. If it is non-NULL, cast **ppDetailsData* to the type defined for your class' details data and fill in the information for this particular instance. If this parameter is NULL, put the required size in *pcp*.

pcp - input

The size of the details data buffer. If *ppDetailsData* is set to NULL, return the size required for this object's details data in *pcp*.

RETURNS

TRUE—The call was successful.

FALSE—An error occurred.

HOW TO CALL

This method is only called by the system to get an object's details data when it is awakened. This data is used for the Details view of the object's folder and can be used for sorting or the Include page.

HOW TO OVERRIDE

Override this method to provide the details data for instances of your subclass. This is only necessary if you have overridden `wpclsQueryDetailsInfo` and are supporting details view columns for your subclass. Call the parent method *first*. If *ppDetailsData* is non-NULL, once the details information is placed in **ppDetailsData*, change **ppDetailsData* so that it points one byte *after* your data in memory. This must be done to enable subclasses to write their data in the appropriate place. The data

written must be in the same format as the information specified in `wpclsQueryDetailsInfo`. When calculating the size, `*pcp`, add the size of your data to the value already entered by the parent class. See the sample code (pg. 138) for more information on overriding this method.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpCnrRefreshDetails` -106, `wpclsQueryDetailsInfo` -123

NOTES

Call `wpCnrRefreshDetails` (Warp only) on an object to cause the system to refresh its details information. This should be done from within any application-defined method that sets instance data used in a details column.

RESTRICTIONS/WARNINGS

- The system will call this method when an object is awakened or to refresh the details data of an object. However, in versions before Warp, there is no public method for causing an object's details to be refreshed. The `WPFileSystem` class supports details view data and will periodically refresh the details data of its objects. If your class is a descendant of `WPFileSystem` and you wish to have the details data refreshed, call one of the file system methods that changes details data. For example, calling `wpSetAttr` will cause `wpQueryDetailsData` to be called for that object.

SAMPLE CODE

See the details data implementation sample at the end of this chapter (pg. 138).

● **`wpQueryIcon` `WPObject` instance method**

Returns the handle for the icon representing an object.

SYNTAX

HPOINTER `wpQueryIcon ()`

PARAMETERS

none

RETURNS

(nonzero value)—The handle to the icon for this object.

NULLHANDLE—An error occurred.

HOW TO CALL

Call this method at any time to obtain the handle for the icon of an object.

HOW TO OVERRIDE

This method does not need to be overridden to change the icon for an object. Override `wpclsQueryIconData` to specify the default icon data for the class. Call `wpSetIconData` and then `wpSaveDeferred` to permanently change the icon data and `wpSetIcon` to temporarily change the icon handle of a specific object.

OTHER INFO

Include file: `wobject.h` and `pmwin.h`

SEE ALSO

`wpclsQueryIconData` -124, `wpclsSetIcon` -130, `wpclsSetIconData` -130, `wpQueryIconData` -115, `wpSaveDeferred` -94, `wpSetIcon` -119, `wpSetIconData` -120

● **wpQueryIconData** **WObject instance method**

Returns the data for the icon representing an object.

SYNTAX

ULONG `wpQueryIconData` (**PICONINFO** *pIconInfo*)

PARAMETERS

pIconInfo - input/output

A pointer to an **ICONINFO** structure (pg. 136) containing the requested icon data. If **NULL** is specified, the required size for the buffer is returned.

RETURNS

(nonzero value)—The size of the icon data for this object.

NULLHANDLE—An error occurred.

HOW TO CALL

Call this method at any time to obtain the icon data for an object. Different classes will return icon data in different formats. The WPProgram class will return the format **ICON_DATA** and the **ICON-INFO** structure will contain a pointer to the data. If the icon was set by the user via the General page of the settings notebook, **WPAbstract** or **WPFileSystem** will use the format **ICON_DATA**. **WPObject** will return the icon information returned from **wpclsQueryIconData**.

HOW TO OVERRIDE

This method does not need to be overridden to change the icon for an object. Override **wpclsQueryIconData** to specify the default icon data for the class. Call **wpSetIconData** and then **wpSaveDeferred** to permanently change the icon data and **wpSetIcon** to temporarily change the icon handle of a specific object.

OTHER INFO

Include file: `wpobject.h` and `os2def.h`

SEE ALSO

wpclsQueryIconData -124, **wpclsSetIcon** -130, **wpclsSetIconData** -130, **wpQueryIcon** -114, **wpSetIcon** -119, **wpSetIconData** -120

SAMPLE CODE

The following sample demonstrates how to call **wpQueryIconData**. This can be done at any time from any method or function. In this example, the variable **Object** is assumed to be a valid **SOMAny *** instance pointer.

```
ULONG ulSize;
PICONINFO pIconInfo;

ulSize = _wpQueryIconData(Object, NULL);
if (ulSize)
{
    pIconInfo = (PICONINFO)_wpAllocMem(Object, ulSize, NULL);
    if (pIconInfo)
```

```
{
    _wpQueryIconData(Object, pIconInfo);
    /* access the data here..... */
    /* and then free it when done */
    _wpFreeMem(Object, (PBYTE) pIconInfo);
}
```

●	wpQueryObjectID	WPObject instance method (Warp only)
---	------------------------	---

Returns the ID string of an object.

SYNTAX

PSZ wpQueryObjectID()

PARAMETERS

none

RETURNS

(nonzero value)—A pointer to the ID string for this object. Do not free this memory.

NULL—This object does not have an object ID.

HOW TO CALL

Call this method on any object to query its object ID.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpobject.h

SEE ALSO

wpSetup -63

NOTES

An object ID is a string that uniquely identifies an object. See Chapter 2 for more information.

RESTRICTIONS/WARNINGS

- The method, wpSetObjectID (Warp only), does not write the object ID to the ini file and, therefore, is not useful. To set the object ID, call wpSetup with the OBJECTID keyword.

- **wpQueryStyle** **WPObject instance method**

Returns the style of an object.

SYNTAX

ULONG wpQueryStyle ()

PARAMETERS

none

RETURNS

The current style flags of the object ORed together. See wpModifyStyle for a list of object style flags.

HOWTO CALL

Call this method at any time to obtain the current styles of an object. Call wpclsQueryStyle to obtain the default style for the class.

HOWTO OVERRIDE

This method is generally not overridden. Call the wpModifyStyle method to change the style flags of an object.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpclsQueryStyle -127, wpModifyStyle -108, wpSetStyle -121

- **wpQueryTitle** **WPObject instance method**

Returns the title of an object.

SYNTAX

PSZ wpQueryTitle ()

PARAMETERS

none

RETURNS

The pointer to the object's title string. Do not free this memory.

HOW TO CALL

Call this method at any time to obtain the current title of an object. Call wpclsQueryTitle to determine the default title for the class.

HOW TO OVERRIDE

This method is generally not overridden. Call the wpSetTitle method to change the title of an object.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpclsQueryTitle -129, wpSetTitle -122

● wpSetIcon WPObject instance method

Sets the handle for the icon representing an object.

SYNTAX

BOOL wpSetIcon (HPOINTER *hptrNewIcon*)

PARAMETERS

hptrNewIcon - input

The icon handle to set for this object.

RETURNS

TRUE—The icon was set successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to set the icon of an object. Changing the icon with `wpSetIcon` is not a persistent change. To permanently change the icon of an object, call `wpSetIconData` and then `wpSaveDeferred`.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.h` and `pmwin.h`

SEE ALSO

`wpclsQueryIconData` -124, `wpclsSetIcon` -130, `wpclsSetIconData` -130, `wpQueryIcon` -114, `wpQueryIconData` -115, `wpSetIconData` -120

● wpSetIconData WPObject instance method

Sets the data for the icon representing an object.

SYNTAX

BOOL `wpSetIconData` (**PICONINFO** *pIconInfo*)

PARAMETERS

pIconInfo - input

A pointer to an **ICONINFO** structure (pg. 136) that contains the information about the new icon for this object.

RETURNS

TRUE—The icon was set successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to change the icon of an object. Changes made by this method can be saved with `wpSaveDeferred`. To temporarily change the icon, call `wpSetIcon`.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wobject.h` and `os2def.h`

SEE ALSO

`wpcQueryIconData` -124, `wpcSetIcon` -130, `wpcSetIconData` -130,
`wpQueryIcon` -114, `wpQueryIconData` -115, `wpSetIcon` -119

● **wpSetStyle** **WObject instance method**

Sets the style of an object.

SYNTAX

BOOL wpSetStyle (**ULONG** *ulNewStyle*)

PARAMETERS

ulNewStyle - input

The new style flags for this object. Multiple style flags can be ORed together at once. See `wpModifyStyle` for a list of flags.

RETURNS

TRUE—The style was successfully changed.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time to change the style of an object. However, `wpModifyStyle` is preferred because it only affects the style flags specified. `wpSetStyle` completely replaces all the style settings with *ulNewStyle*.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wobject.h`

SEE ALSO

`wpcQueryStyle` -127, `wpModifyStyle` -108, `wpQueryStyle` -118

● wpSetTitle WPObject instance method

Sets the title of an object.

SYNTAX

BOOL wpSetTitle (**PSZ** *pszNewTitle*)

PARAMETERS

pszNewTitle - input

A pointer to the new title for the object.

RETURNS

TRUE—The title was successfully set.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to set the title for an object. There is no limit for the size of the string as long as there is enough memory in the system. Workplace will make a copy of the string; therefore, it can be freed after the call is made.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpclQueryTitle` -129, `wpQueryTitle` -118

Class Methods

wpclQueryDetailsInfo returns the details column information for a class (pg. 123).

wpclQueryIconData returns the data for the default icon for a class (pg. 124).

wpclsQuerySetting (Warp) returns a setting from a device object (pg. 126).

wpclsQueryStyle returns the default style for a class (pg. 127).

wpclsQueryTitle returns the default title for a class (pg. 129).

wpclsSetIcon (Warp) sets the icon for a class (pg. 130).

wpclsSetIconData (Warp) sets the icon for a class (pg. 130).

wpclsSetSetting (Warp) sets the value for a device object setting (pg. 131).

● **wpclsQueryDetailsInfo** **WPObject class method**

Returns the details column information for a class.

SYNTAX

ULONG wpclsQueryDetailsInfo (**PCLASSFIELDINFO** **ppClassFieldInfo*, **PULONG** *pSize*)

PARAMETERS

ppClassFieldInfo - input

This pointer contains a pointer to a linked list of **CLASSFIELDINFO** structures (pg. 132). Each class defines a node in the linked list for every details column that is to be implemented. This memory should be allocated and initialized in **wpclsInitData**; or, a global array of **CLASSFIELDINFO** structures can be used.

pSize - input

The sum of the required sizes for the details data buffers defined by the class and parent classes.

RETURNS

The sum of the number of details columns defined by the class and parent classes.

HOW TO CALL

This method is only called by the system to obtain the details column information for a class and all of the classes in the parent chain. This data is used for the Details view of the object's folder and can be used for sorting or the Include page.

HOWTO OVERRIDE

Override this method to provide details data column information for your subclass. Call the parent method *first*. If *ppClassFieldInfo* is non-NULL, add the CLASSFIELDINFO structures for this class at the end of the linked list, **ppClassFieldInfo*. If *pSize* is non-NULL, add the size of the class-defined details data structure to the size already placed in **pSize* by the parent classes. Return the number of CLASSFIELDINFO nodes in the **ppClassFieldInfo* list for *this* class, plus the number returned when the parent method was called. See the sample code (pg. 138) for more information on overriding this method.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpQueryDetailsData` -113

SAMPLE CODE

See the details data implementation sample at the end of this chapter (pg. 138).

● wpclsQueryIconData WPObject class method

Returns the data for the default icon for a class.

SYNTAX

ULONG wpclsQueryIconData (**PICONINFO** *pIconInfo*)

PARAMETERS

pIconInfo - input/output

A pointer to an ICONINFO structure (pg. 136) containing the requested icon data. If NULL is specified, the required size for the buffer is returned.

RETURNS

(nonzero value)—The size of the icon data for this class.

0—An error occurred.

HOWTO CALL

Call this method at any time to obtain the icon data for a class. This method is called by the system to determine the default icon for objects.

HOWTO OVERRIDE

Override this method to specify the default icon data for your class. If *pIconInfo* is NULL, just return the required size for the data. If the ICON_DATA format will be used, include the size of the pIconInfo->pIconData buffer as well. If *pIconInfo* is non-NULL, fill in the icon data for the class and return the total size. If the data in *pIconInfo* will be in the ICON_DATA format, set pIconInfo->pIconData to (PVOID)(pIconInfo+1). You do not need to call the parent method.

OTHER INFO

Include file: wobject.h and os2def.h

SEE ALSO

wpclsSetIcon -130, wpclsSetIconData -130, wpQueryIcon -114,
wpQueryIconData -115, wpSetIcon -119, wpSetIconData -120

SAMPLE CODE

The following sample demonstrates how to override wpclsQueryIconData for the class MyClass. The most straightforward way to implement this override is using the ICON_RESOURCE format.

```
SOM_Scope ULONG   SOMLINK mclsM_wpclsQueryIconData(M_MyClass *somSelf,
                                                    PICONINFO pIconInfo)
{
    PSZ   pszPathName;
    HMODULE hmod;
    somId idClass;

    idClass = SOM_IdFromString("MyClass");
    pszPathName = _somLocateClassFile( SOMClassMgrObject,
                                       idClass,
                                       MyClass_MajorVersion,
                                       MyClass_MinorVersion);

    SOMFree(idClass);
    DosQueryModuleHandle( pszPathName, &hmod);

    /* test if pIconInfo is NULL in case this call is just to query
       the size */
    if (pIconInfo)
    {
        pIconInfo->cb = sizeof(ICONINFO);
        pIconInfo->fFormat = ICON_RESOURCE;
    }
}
```

```

    pIconInfo->pszFileName = 0L;
    pIconInfo->hmod = hmod;    // module in which to find the
                                // resource
    pIconInfo->resid = 1000;    // resource id for the icon
    pIconInfo->cbIconData = 0L;
    pIconInfo->pIconData = NULL;
} /* endif */
return sizeof(ICONINFO);
}

```

● wpclsQuerySetting WPAbstract class method (Warp only)

Returns a setting from a device object.

SYNTAX

ULONG wpclsQuerySetting (**PSZ** *pszSetting*, **PVOID** *pValue*, **ULONG** *ulValueLen*)

PARAMETERS

pszSetting - input

A pointer to a string containing the name of the setting to query. See Appendix F for a list of device objects that support this method as well as the setting strings they have defined.

pValue - input/output

A pointer to the buffer that will hold the returned setting. If set to NULL, the required size is returned.

ulValueLen - input

The size of the *pValue* buffer.

RETURNS

(nonzero value)—The size of the data for this setting.

0—An error occurred and no data was returned.

HOW TO CALL

Call this method at any time to query a setting for a WPAbstract subclass.

HOW TO OVERRIDE

WPAbstract subclasses can override this method to provide their own settings values. Check the *pszSetting* parameter, and if it is not a string

defined by your subclass, call the parent method. Otherwise, check *ulValueLen* to be sure the buffer is the correct size, place the setting value in *pValue*, and return the size of the data.

OTHER INFO

Include file: wpabs.h

SEE ALSO

wpclsSetSetting -131

● wpclsQueryStyle WPObject class method

Returns the default style for a class.

SYNTAX

ULONG wpclsQueryStyle ()

PARAMETERS

none

RETURNS

The current class style flags of the class ORed together. The following class styles correspond to the object styles that are used as the default style for all objects of this class:

Constant		Corresponding object style
CLSSTYLE_NEVERMOVE	0x02	OBJSTYLE_NOMOVE
CLSSTYLE_NEVERLINK	0x04	OBJSTYLE_NOLINK
CLSSTYLE_NEVERCOPY	0x08	OBJSTYLE_NOCOPY
CLSSTYLE_NEVERDELETE	0x40	OBJSTYLE_NODELETE
CLSSTYLE_NEVERPRINT	0x80	OBJSTYLE_NOPRINT
CLSSTYLE_NEVERDRAG	0x100	OBJSTYLE_NODRAG
CLSSTYLE_NEVERVISIBLE	0x200	OBJSTYLE_NOTVISIBLE

CLSSTYLE_NEVERTEMPLATE (or CLSSTYLE_DONTTEMPLATE in Warp) prevents the user from making an object a template. The Template checkbox is removed from the General page of the settings notebook and default templates are not created in the Templates folder for the class.

CLSSTYLE_PRIVATE (or CLSSTYLE_HIDDEN in Warp) prevents the class from being displayed in class lists in the user interface (for example, the Find dialog).

HOW TO CALL

Call this method at any time to obtain the default styles for a class.

HOW TO OVERRIDE

Override this method to change the default styles for a class. Call the parent first and store the return code. Take out or add any flags from the parent's returned styles.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpModifyStyle -108, wpQueryStyle -118, wpSetStyle -121

NOTES

Call wpModifyStyle to change the style flags of an individual object.

RESTRICTIONS/WARNINGS

- When an object is created, wpclsQueryStyle is called to query the default styles for the object. This information is stored in the persistent instance data of the object, and wpclsQueryStyle is not called again for this object. If the class implementation is then modified and the class default style changes, it will not affect objects that have already been created.
- The default styles of the parent of your class may not be desirable. For example, the WPAbstract class, by default, has CLSSTYLE_NEVERTEMPLATE. After creating the first instance of your class, be sure to examine the behavior of the object and change any of the style flags in your wpclsQueryStyle override to remove unwanted behavior from the parent class.

SAMPLE CODE

The following sample demonstrates an override of wpclsQueryStyle from the class MyClass. MyClass is a subclass of WPAbstract but has templateable objects.

```
SOM_Scope ULONG    SOMLINK mclsM_wpclsQueryStyle(M_MyClass *somSelf)
{
    /* call the parent method and take out the CLSSTYLE_NEVERTEMPLATE
       flag */
    return (parent_wpclsQueryStyle(somSelf)
           & ~ CLSSTYLE_NEVERTEMPLATE);
}
```

● **wpclsQueryTitle** **WPObject class method**

Returns the default title for a class.

SYNTAX

PSZ wpclsQueryTitle ()

PARAMETERS

none

RETURNS

The pointer to the class' title string.

HOW TO CALL

Call this method at any time to obtain the default title for a class. Do not free the returned string.

HOW TO OVERRIDE

Override this method to provide the default title for your class. Do not call the parent method. The string returned from your override should not be freed while the class is loaded. It is easiest to create a global variable that is an array of characters and return it from the wpclsQueryTitle override.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpQueryTitle -118, wpSetTitle -122

● wpclsSetIcon WPObject class method (Warp only)

Sets the default icon for a class.

SYNTAX

BOOL wpclsSetIcon(**HPOINTER** *hptrNewIcon*)

PARAMETERS

hptrNewIcon - input

The icon handle to set for this class.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to set the default icon for a class. This change is only temporary and will only last until the class is unloaded. Usually it is best to override wpclsQueryIconData instead of calling this method.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpclsQueryIconData -124, wpclsSetIconData -130, wpQueryIcon -114, wpQueryIconData -115, wpSetIcon -119, wpSetIconData -120

● wpclsSetIconData WPObject class method (Warp only)

Sets the default icon for a class.

SYNTAX

BOOL wpclsSetIconData(**PICONINFO** *pIconInfo*)

PARAMETERS

pIconInfo - input

A pointer to an ICONINFO structure (pg. 136) containing the icon data to set on the class. flFormat must be ICON_RESOURCE.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to set the default icon for a class. This change is only temporary and will only last until the class is unloaded. Usually it is best to override wpclsQueryIconData instead of calling this method.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpclsQueryIconData -124, wpclsSetIcon -130, wpQueryIcon -114, wpQueryIconData -114, wpSetIcon -119, wpSetIconData -120

● wpclsSetSetting WPAbstract class method (Warp only)

Sets the value for a device object setting.

SYNTAX

BOOL wpclsSetSetting (**PSZ** *pszSetting*, **PVOID** *pValue*)

PARAMETERS

pszSetting - input

A pointer to a string containing the name of the setting. See Appendix F for a list of device objects that support this method as well as the setting strings they have defined.

pValue - input

A pointer to the buffer that contains the data to set.

RETURNS

TRUE—The value was set successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to set a settings value for a WPAbstract subclass.

HOW TO OVERRIDE

WPAbstract subclasses can override this method to provide their own settings values. Check the *pszSetting* parameter, and if it is not a string defined by your subclass, call the parent method. Otherwise, change the value of the specified setting to the new value passed in *pValue*.

OTHER INFO

Include file: wpabs.h

SEE ALSO

wpclsQuerySetting -126

General Object Settings: Structures

CLASSFIELDINFO**(80)**

cb (ULONG) 0

The size of the structure, in bytes, including cb itself.

flData (ULONG) 4

Attributes of the field's data. This is the same as the flData field in the FIELDINFO structure for container details view. OR together multiple flags to specify the attributes of the data displayed in a column.

Constant		Description
CFA_LEFT	0x001	Left align the text.
CFA_RIGHT	0x002	Right align the text.
CFA_CENTER	0x004	Center the text.

CFA_TOP	0x008	Top align the text.
CFA_VCENTER	0x010	Vertically center the text.
CFA_BOTTOM	0x020	Bottom align the text.
CFA_INVISIBLE	0x040	Make this column invisible.
CFA_BITMAPORICON	0x100	Data is a bitmap or an icon handle.
CFA_SEPARATOR	0x200	A vertical separator will be placed to the right of this data.
CFA_HORZSEPARATOR	0x400	A horizontal separator will be placed below this data.
CFA_STRING	0x800	Data is a pointer to a string of characters.
CFA_OWNER	0x1000	Ownerdraw field.
CFA_DATE	0x2000	Data is a pointer to a CDATE structure.
CFA_TIME	0x4000	Data is a pointer to a CTIME structure.
CFA_FIREADONLY	0x8000	Column is read-only.
CFA_FITITLE READONLY	0x10000	Column title is read-only (use this flag for flTitle).
CFA_ULONG	0x20000	Column is in number format.

flTitle (ULONG)	8
Attributes of the field's title. This is the same as the flTitle field in the FIELDINFO structure for container details view. OR together multiple flags to specify the attributes of the title for a column. See flData for a list of flags.	
pTitleData (PVOID)	12
The title data. The flTitle flags specify what type of data this is.	
ulReserved (ULONG)	16
Reserved by the system. Set to 0.	
pUserData (PVOID)	20
Pointer to optional user data.	
pNextFieldInfo (PCLASSFIELDINFO)	24
Pointer to the next CLASSFIELDINFO structure in the linked list.	
cxWidth (ULONG)	28
The width of the field in pixels. Can be 0.	

offFieldData (ULONG)

32

Offset from the beginning of this class' data for this field. For example, if the class defines the following details data:

```
typedef struct _MYDETAILS
{
    PSZ pszString;
    ULONG ulNumber;
} MYDETAILS;
```

in the following CLASSFIELDINFO array, offFieldData would be filled in as

```
CLASSFIELDINFO ClassFieldInfo<[>2];

ClassFieldInfo[0].offFieldData = FIELDOFFSET(MYDETAILS, pszString);
ClassFieldInfo[1].offFieldData = FIELDOFFSET(MYDETAILS, ulNumber);
```

Note: The order of the field definition structures in the linked list must be the same as the order of fields in the details data structure.

ulLenFieldData (ULONG)

36

The size of the details data in bytes. Using the sample example for offFieldData,

```
ClassFieldInfo[0].ulLenFieldData = sizeof(PSZ);
ClassFieldInfo[1].ulLenFieldData = sizeof(ULONG);
```

pfnOwnerDraw (PFNOWNDRW)

40

The ownerdraw procedure for this column. Can be NULL unless CFA_OWNER was specified for this column. PFNOWNDRW functions should be defined as follows:

```
BOOL EXPENTRY OwnerDrawFunc(HWND hwnd, PVOID Param1, PVOID Param2);
```

hwnd is the handle of the window to be painted. **Param1** is a pointer to an OWNERITEM structure used for owner draw containers. **Param2** is a pointer to the value to draw. This is NULL except when this function is called to paint details displayed in the Include criteria dialog.

Return TRUE if the field was ownerdrawn and FALSE if the container control should do the default painting.

flCompare (ULONG)

44

The following flags can be ORed together to specify the sort and include attributes for this field:

Constant		Description
COMPARE_SUPPORTED	0x01	These values can be used with comparison functions to support the Include and Sort page.
SORTBY_SUPPORTED	0x02	This column can be used to sort the folder views.

pfnCompare (PFNCOMPARE)

48

Comparison function for this field. This value allows you to specify your own function to be called when two values for this details column are compared for sort or include support. Set to NULL if COMPARE_SUPPORTED was not specified in the flCompare flags. Compare functions should be defined as follows:

```
LONG EXPENTRY CompareValues(PVOID Param1, PVOID Param2);
```

Param1 is a pointer to the details field value. **Param2** is a pointer to the value to be compared with.

Return one of the following values:

Constant		Description
CMP_EQUAL	0	The two values are equal.
CMP_GREATER	1	Param1 is greater than Param2.
CMP_LESS	2	Param1 is less than Param2.

DefaultComparison (ULONG)

52

The default compare operator in the include page add/change criteria dialog.

Constant		Description
CMP_EQUAL	0	Show the object if it is equal to the compare value.
CMP_GREATER	1	Show the object if it is greater than the compare value.
CMP_LESS	2	Show the object if it is less than the compare value.
CMP_GREATER_OR_EQUAL	3	Show the object if it is greater than or equal to the compare value.
CMP_LESS_OR_EQUAL	4	Show the object if it is less than or equal to the compare value.
CMP_NOT_EQUAL	5	Show the object if it is not equal to the compare value.

ulLenCompareValue (ULONG)

56

Maximum length of the compare data. This must be filled in if COMPARE_SUPPORTED is specified in flCompare.

pDefCompareValue (PVOID)

60

The default value for comparisons in the Include page criteria Add/Change dialog. Can be NULL unless *ownerdraw* is used for this column.

pMinCompareValue (PVOID) 64

The minimum compare value. This value will be used to set the limits of the spin button for CFA_ULONG details data on the Include page criteria Add/Change dialog. Can be NULL.

pMaxCompareValue (PVOID) 68

The maximum compare value. This value will be used to set the limits of the spin button for CFA_ULONG details data on the Include page criteria Add/Change dialog. Can be NULL.

pszEditControlClass (PSZ) 72

A pointer to a string containing the name of the class for a user-defined edit control. The support for user-defined edit controls for the Include page Add/Change criteria dialog currently does not work. Therefore, this field should be set to NULL and flCompare should be zero if this column is CFA_OWNER.

pfnSort (PFNCOMPARE) 76

The function for this field used to sort data from this column. This value can be NULL. See pfnCompare for a description of an FNCOMPARE function.

Note: If you specify SORTBY_SUPPORTED in the flCompare flags and the type of data for this column is CFA_ULONG, the default sort behavior is to sort numbers in descending order. You must provide a sort function for this column if you want your values to sort in ascending order.

Used by: wpclsQueryDetailsInfo -123

ICONINFO**28****cb (ULONG)** 0

The size in bytes of the ICONINFO structure including cb itself.

fFormat (ULONG) 4

The format of the data contained in the ICONINFO structure. Specify one of the following:

Constant		Description
ICON_FILE	1	This icon data is a file name, and pszFileName contains the fully qualified path of an .ico file.
ICON_RESOURCE	2	This icon data is an icon resource. hmod contains the module name, and resid contains the resource ID for this icon.

ICON_DATA	3	This icon data is in raw data format. <code>cbIconData</code> contains the size of the icon data, and <code>pIconData</code> points to the bitmap data for this icon.
ICON_CLEAR	4	Set <code>flFormat</code> to <code>ICON_CLEAR</code> when <i>calling</i> <code>wpSetIconData</code> to have the object's icon revert back to its default. If the object is a <code>WPPProgram</code> , it will use the icon for the executable the object refers to.

pszFileName (PSZ) 8

The fully qualified path of a .ico file. This value is ignored if `flFormat` is not `ICON_FILE`.

hmod (HMODULE) 12

The handle for the module that contains the icon resource specified by `resid`. This value is ignored if `flFormat` is not `ICON_RESOURCE`.

resid (ULONG) 16

The ID for the icon resource contained in `hmod`. This value is ignored if `flFormat` is not `ICON_RESOURCE`.

cbIconData (ULONG) 20

This size of the icon data that `pIconData` points to. This value is ignored if `flFormat` is not `ICON_DATA`.

pIconData (PVOID) 24

A pointer to the bitmap data for this icon. This value is ignored if `flFormat` is not `ICON_DATA`.

Used by: `wpclsQueryIconData` -124, `wpclsSetIconData` -130, `wpQueryIconData` -115, `wpSetIconData` -120

POINTL

(8)

x (LONG) 0

The x coordinate for a point.

y (LONG) 4

The y coordinate for a point.

Used by: `wpCnrInsertObject` -233, `wpQueryNextIconPos` -251

Details Data Sample Code

The following sample demonstrates how to implement details data for an object class. The relevant code from the .C and .H files for the class MyClass have been included. The class methods are shown before the instance methods to make this easier to follow.

From the .h file:

```
HMODULE hmod;      // handle of the class .DLL Set in wpclsInitData...
#define MAXSTRING 256
#define NUM_DETAILS_FIELDS 2  // number of details view columns for
                               // this class
CLASSFIELDINFO MyClassFieldInfo<[>NUM_DETAILS_FIELDS]; //Details
                                                    //column
                                                    //definitions

/* below is the class specific details data. MyClassFieldInfo is an
   array of
   * structures that define the columns. The class only has ONE
   definition
   * structure. Each object will have extra memory to hold a
   MYCLASSDATA structure.
   * This structure is filled in for each object in the
   wpQueryDetailsData override.
   */
typedef struct _MYCLASSDATA
{
    PSZ pszComment;           //will point to _pszComment instance data
    ULONG ulHandle;           //the object handle. Not very useful data
                               //but demonstrates numeric details data.
} MYCLASSDATA, *PMYCLASSDATA;

LONG EXPENTRY CompareHandles(PVOID Param1, PVOID Param2);
LONG EXPENTRY CompareStrings(PVOID Param1, PVOID Param2);
MRESULT EXPENTRY EditControlProc(HWND hwnd, ULONG msg, MPARAM mp1,
MPARAM mp2);

/* the following three ULONGs are used to define the Handles details
   data */
ULONG ulMinValue = 0;      // the minimum value for comparison criteria
ULONG ulMaxValue = 5;      // the maximum value for comparison criteria
ULONG ulDefValue = 3;      // the default value for comparison criteria
```


From the .c file:

```
#undef SOM_CurrentClass
#define SOM_CurrentClass SOMMeta

/* override wpclsInitData to initialize the global MyClassFieldInfo
   array and set the hmod global variable.
*/
SOM_Scope void  SOMLINK mclsM_wpclsInitData(M_MyClass *somSelf)
{
    PSZ  pszPathName;
    PCLASSFIELDINFO pInfo;
    somId idClass;

    parent_wpclsInitData(somSelf);
    idClass = SOM_IdFromString("MyClass");
    pszPathName = _somLocateClassFile(  SOMClassMgrObject,
                                        idClass,
                                        MyClass_MajorVersion,
                                        MyClass_MinorVersion);

    SOMFree(idClass);
    DosQueryModuleHandle( pszPathName, &hmod);

    /* Setup the MyClassFieldInfo array. Use the pInfo pointer to make
       * referencing easier.
       */

    /* point pInfo to the first item in the MyClassFieldInfo array which
       * is the Comment column. The structures in this array must be in the
       * same order as the fields in the MYCLASSDATA structure that they
       * represent.
       */
    pInfo = MyClassFieldInfo;

    /* zero out the structure */
    memset((PCH) pInfo, 0, sizeof(CLASSFIELDINFO));
    pInfo->cb = sizeof(CLASSFIELDINFO);

    /* the data in this column will be a string so use CFA_STRING*/
    pInfo->flData = CFA_RIGHT | CFA_SEPARATOR | CFA_FIREADONLY |
                   CFA_STRING;
```

```

pInfo->flTitle = CFA_CENTER | CFA_SEPARATOR | CFA_HORZSEPARATOR |
                CFA_STRING | CFA_FITITLEREADONLY;

/* pInfo+1 is the next field info structure in the array */
pInfo->pNextFieldInfo = pInfo+1;
pInfo->pTitleData = "Comment";
pInfo->flCompare = COMPARE_SUPPORTED | SORTBY_SUPPORTED;
pInfo->offFieldData = FIELDOFFSET(MYCLASSDATA, pszComment);
pInfo->ulLenFieldData = sizeof(PSZ);
pInfo->DefaultComparison = CMP_EQUAL;
pInfo->pfnCompare = CompareStrings; // provide a user-defined compare
                                // function
pInfo->pfnSort = CompareStrings; // use the same function for sort
pInfo->ulLenCompareValue = MAXSTRING; // this value is required.

/* point pInfo to the next field info structure */
pInfo++;
memset((PCH) pInfo, 0, sizeof(CLASSFIELDINFO));
pInfo->cb = sizeof(CLASSFIELDINFO);

/* the data for this column is numeric so use CFA_ULONG */
pInfo->flData = CFA_RIGHT | CFA_SEPARATOR | CFA_FIREADONLY |
               CFA_ULONG;
pInfo->flTitle = CFA_CENTER | CFA_SEPARATOR | CFA_HORZSEPARATOR |
                CFA_STRING | CFA_FITITLEREADONLY;
pInfo->pNextFieldInfo = NULL;
pInfo->pTitleData = "Handle";
pInfo->flCompare = COMPARE_SUPPORTED | SORTBY_SUPPORTED;
pInfo->offFieldData = FIELDOFFSET(MYCLASSDATA, ulHandle);
pInfo->ulLenFieldData = sizeof(ULONG);
pInfo->ulLenCompareValue = sizeof(ULONG);
pInfo->DefaultComparison = CMP_EQUAL;
pInfo->pfnCompare = CompareHandles;
pInfo->pfnSort = CompareHandles;
pInfo->pMinCompareValue = (PVOID)&ulMinValue;
pInfo->pMaxCompareValue = (PVOID)&ulMaxValue;
pInfo->pDefCompareValue = (PVOID)&ulDefValue;
}

/* override wpclsQueryDetailsInfo to add the field info structures
 * for this class to the linked list.
 */

```

```
SOM_Scope ULONG  SOMLINK mclsM_wpclsQueryDetailsInfo(M_MyClass* somSelf,
                                                       PCLASSFIELDINFO
                                                       *ppClassFieldInfo,
                                                       PULONG pSize)
{
    ULONG ulNumColumns, i;
    PCLASSFIELDINFO pInfo;
    /* call the parent class FIRST */
    ulNumColumns = parent_wpclsQueryDetailsInfo
        (somSelf, ppClassFieldInfo, pSize);

    /* if the pSize pointer is set, add the size of our class specific
     * data to the size already specified by the parent classes.
     */
    if (pSize)
        *pSize = *pSize + sizeof(MYCLASSDATA);

    /* if the ppClassFieldInfo pointer is set, put our details info
     * structures at the end of the linked list.
     */
    if (ppClassFieldInfo)
    {
        if (*ppClassFieldInfo)
        {
            pInfo = *ppClassFieldInfo;

            /* loop through the ppClassFieldInfo list and find the end */
            for (i = 1; i < ulNumColumns; i++)
            {
                if (pInfo->pNextFieldInfo)
                    pInfo = pInfo->pNextFieldInfo;
            }
            /* pInfo now points to the end. Set its next pointer to the
             * first node in our list (MyClassFieldInfo). Note that when we
             * initialized this structure in wpclsInitData, we already set
             * the next pointers of the nodes.
             */
            pInfo->pNextFieldInfo = MyClassFieldInfo;
        }
        else
        {
            /* there are no fields in the list. Put ours at the head. */
            *ppClassFieldInfo = MyClassFieldInfo;
        }
    }
}
```

```

        /* return the number of columns returned from the parent classes
           plus the number
           * defined by this class.
           */
        return(ulNumColumns + NUM_DETAILS_FIELDS);
    }
#undef SOM_CurrentClass
#define SOM_CurrentClass SOMInstance

/* override wpQueryDetailsData to fill in the details information for
   * a specific instance of this class.
   */
SOM_Scope ULONG  SOMLINK mcls_wpQueryDetailsData(MyClass *somSelf,
                                                    PVOID *ppDetailsData,
                                                    PULONG pcp)
{
    PMYCLASSDATA pMyDetails;
    MyClassData *somThis = MyClassGetData(somSelf);

    /* call the parent method first */
    parent_wpQueryDetailsData(somSelf, ppDetailsData, pcp);
    /* if ppDetailsData is set, it has already been modified by the
       * parent class to point to where our data must be written.
       */
    if (ppDetailsData)
    {
        /* point pMyDetails to where the data will be written and fill
           * in the details data.
           */
        pMyDetails = (PMYCLASSDATA) *ppDetailsData;
        pMyDetails->pszComment = _pszComment ? _pszComment : NULL;
        pMyDetails->ulHandle = _wpQueryHandle(somSelf);

        /* point ppDetailsData to location after my data */
        *ppDetailsData = ((PBYTE) (*ppDetailsData)) +
            sizeof(MYCLASSDATA);
    }
    if (pcp)
        /* add the size of the buffer to *pcp */
        *pcp += sizeof(MYCLASSDATA);
    return(TRUE);
}

```

```
#undef SOM_CurrentClass
/* CompareHandles is the sort and include page comparison function
 * for the ulHandle details data. It is not necessary to provide a
 * compare function but in this case, sort will by default sort
 * numbers in descending order unless a sort function is provided.
 */
LONG EXPENTRY CompareHandles(PVOID Param1, PVOID Param2)
{
    ULONG ulValue, ulCompare;

    ulValue = *((PULONG)Param1);
    ulCompare = *((PULONG)Param2);

    if (ulValue < ulCompare)
        return(CMP_LESS);
    else if (ulValue == ulCompare)
        return(CMP_EQUAL);

    else
        return(CMP_GREATER);
}

/* CompareStrings is the sort and include page comparison function
 * for the pszComment details data. It is not necessary to provide
 * a compare function for string data. (it is here for
 * demonstration purposes)
 */
LONG EXPENTRY CompareStrings(PVOID Param1, PVOID Param2)
{
    PSZ pszValue, pszCompare;
    LONG lRet;

    pszValue = (PSZ) Param1;
    pszCompare = (PSZ) Param2;

    lRet = strcmp(pszValue, pszCompare);
    if (lRet < 0)
        return(CMP_LESS);
    else if (lRet > 0)
        return(CMP_GREATER);
    else
        return(CMP_EQUAL);
}
```

6

In-Use List

Each Workplace object has an in-use list defined by the `WPObj` class. The in-use list is a linked list of `USEITEM` (pg. 154) structures that contain information on resources currently in use by the object. When an object opens a view, allocates memory, is added to a container view, opens an association, or has awakened shadows, the system will add an item to the in-use list which can later be accessed by calling `wpFindUseItem`. It is the programmer's responsibility to add `USAGE_OPENVIEW` use items to the in-use list, using `wpAddToObjUseList`, whenever an application-defined view is created. The system then uses this information to place hash marks on the object's icon, to switch to existing views, and to close views of an object when it is deleted.

The `USEITEM` structures are always followed immediately in memory by structures specific to the type of item. When adding an item to the in-use list, allocate the `USEITEM` structure and the type specific structures together to ensure that they will follow one another in memory.

Restrictions and Warnings

- Applications cannot define new USEITEM structure types. Only the predefined USAGE_* constants are acceptable.

Instance Methods

wpAddToObjUseList adds an item to the in-use list of an object (pg. 145).

wpDeleteFromObjUseList removes an item from the in-use list of an object (pg. 148).

wpFindUseItem finds an item in the in-use list of an object (pg. 150).

wpFindViewItem (Warp) finds a VIEWITEM in the in-use list of an object. (pg. 152).

● **wpAddToObjUseList** **WPObj** instance method

Adds an item to the in-use list of an object.

SYNTAX

BOOL wpAddToObjUseList (**PUSEITEM** *pUseItem*)

PARAMETERS

pUseItem - input

The address of the USEITEM (pg. 154) to be added to the in-use list. A type dependent structure must immediately follow the USEITEM structure in memory. Do not free *pUseItem* after calling this method, as the system does not make a copy of the memory that *pUseItem* points to.

RETURNS

TRUE—The item was successfully added to the in-use list.

FALSE—An error occurred.

HOW TO CALL

Call this method on any object to add an item to the in-use list. The system will lock the object once if the type of use item is `USAGE_RECORD`, `USAGE_LINK`, or `USAGE_OPENVIEW`. Applications that define new views for objects should add `USAGE_OPENVIEW` items to the in-use list when the view is created. These items should be removed with the `wpDeleteFromObjUseList` method when the view closes.

HOW TO OVERRIDE

It is not generally necessary to override this method in a Workplace application, but it can be used as a notification of when items are added to the in-use list. Be sure to call the parent method. Use a `wpOpen` override instead for notification when a view is opened.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpAllocMem` -80, `wpCreateShadowObject` -50,
`wpDeleteFromObjUseList` -148, `wpFindUseItem` -150, `wpOpen` -160,
`wpSwitchTo` -173

NOTES

`wpAddToObjUseList` is predominantly called by applications to add `USAGE_OPENVIEW` items to the in-use list. Other types of `USEITEM` structures can be added when applicable, but are usually added by the system. The system will call this method to add a `USAGE_LINK` item when a shadow is created or to add `USAGE_MEMORY` items when memory is allocated with `wpAllocMem`. `USAGE_RECORD` items are added for an object each time it is added to a container, and `USAGE_OPENFILE` items are added for a data file when it opens a program as its view.

RESTRICTIONS/WARNINGS

- The memory for the type-dependent structure must immediately follow that of the `USEITEM` structure; therefore, they should be allocated together.

SAMPLE CODE

The following sample demonstrates how to add an item to the in-use list. In this example, a `USAGE_OPENVIEW` use item is added from the

window procedure itself. The item could also have been added from the `wpOpen` override, but it is easier for the window procedure to keep track of the use item so it can be removed from the in-use list when the view is closed.

```
typedef struct
{
    SOMAny *Object;
    USEITEM UseItem; //note that UseItem and ViewItem are next to
                    //each other
    VIEWITEM ViewItem;
} DLGDATA, *PDLGDATA;
```

```
#define OPEN_HELLO OPEN_USER
```

This view is a dialog that was loaded using `WinLoadDlg`, and the `WM_INITDLG` processing from the dialog procedure is shown below:

```
case WM_INITDLG:
{
    SOMAny *Object;
    PDLGDATA pDlgData;

    /* somSelf was passed to the dialog when it was loaded */
    Object = (SOMAny *) mp2;
    pDlgData = (PDLGDATA) _wpAllocMem(Object, sizeof(DLGDATA), NULL);
    if (pDlgData)
    {
        /* hwndDlg is the HWND parameter of this dialog window
           procedure */
        WinSetWindowULong(hwndDlg, QWL_USER, (ULONG) pDlgData);
        memset(pDlgData, 0, sizeof(DLGDATA) );

        pDlgData->Object = Object;
        /* for convenience, the UseItem and ViewItem structures are
           part of the window words.
           */
        pDlgData->UseItem.type = USAGE_OPENVIEW;
        pDlgData->ViewItem.view = OPEN_HELLO;
        pDlgData->ViewItem.handle = hwndDlg; // hwndDlg is the frame

        _wpAddToObjUseList(Object, &pDlgData->UseItem);
    }
}
```

```

    else
        /* Data allocation failed. Destroy the dialog.
        WinDestroyWindow(hwndDlg);
    }
    return(MRESULT) FALSE;

```

● **wpDeleteFromObjUseList** **WPObj** instance method

Removes an item from the in-use list of an object.

SYNTAX

BOOL wpDeleteFromObjUseList (**PUSEITEM** *pUseItem*)

PARAMETERS

pUseItem - input

The address of the USEITEM (pg. 154) to be removed from the in-use list.

RETURNS

TRUE—The item was successfully removed from the in-use list.

FALSE—The item was not found or an error occurred.

HOWTO CALL

Call this method on any object to remove an item from its in-use list. The system will unlock the object once if the type of use item is **USAGE_RECORD**, **USAGE_LINK**, or **USAGE_OPENVIEW**. Applications that add use items using **wpAddToObjUseList** should call this method to remove them when they are no longer valid.

HOWTO OVERRIDE

If a view was defined by a parent class, overriding **wpDeleteFromObjUseList** and checking for **USAGE_OPENVIEW** items is a good notification for when these views are closed. Be sure to call the parent method.

OTHER INFO

Include file: **wpobject.h**

SEE ALSO

wpAddToObjUseList -145, **wpFindUseItem** -150, **wpFreeMem** -82

RESTRICTIONS/WARNINGS

- When `wpDeleteFromObjUseList` is called, the system does not free the memory that was used for the item removed. It is the application's responsibility to free this memory.

SAMPLE CODE

The following sample demonstrates how to remove an item from the in-use list. In this example, it is removed from within the dialog's window procedure of the object's view. The same dialog is used in the sample for `wpAddToObjUseList`.

```
typedef struct
{
    SOMAny *Object;
    USEITEM UseItem; //note that UseItem and ViewItem are next to
                    //each other
    VIEWITEM ViewItem;
} DLGDATA, *PDLGDATA;
```

The following is the `WM_DESTROY` processing in the dialog window procedure:

```
case WM_DESTROY:
{
    PDLGDATA pDlgData;

    /* hwndDlg is the window handle parameter for this dialog window
       procedure */
    pDlgData = (PDLGDATA) WinQueryWindowULong(hwndDlg, QWL_USER);
    if (pDlgData)
    {
        /* remove the use item from the in-use list */
        _wpDeleteFromObjUseList(pDlgData->Object, &pDlgData->UseItem);

        /* free the dialog data */
        _wpFreeMem(pDlgData->Object, (PBYTE) pDlgData);
    }
}
break;
```

● **wpFindUseItem** **WPObj** instance method

Finds an item in the in-use list of an object.

SYNTAX

PUSEITEM wpFindUseItem (**ULONG** *type*, **PUSEITEM** *pCurrentItem*)

PARAMETERS

type - input

The type of the USEITEM to find. Valid types are as follows:

Define	Description
USAGE_MEMORY	1 Usually added by wpAllocMem and is followed by a MEMORYITEM structure.
USAGE_RECORD	4 Added when the object is placed in a container and is followed by a RECORDITEM structure.
USAGE_OPENVIEW	5 Added when a view of an object is opened and is followed by a VIEWITEM structure.
USAGE_LINK	6 Added when a shadow of an object is created and is followed by a LINKITEM structure.
USAGE_OPENFILE	20 Added when a data file opens a program and is followed by a VIEWFILE structure.

The definition of USEITEM and all the item-specific structures are at the end of this chapter.

pCurrentItem -input

The address of an item in the list after which to start the search. If NULL, the search will start with the first item in the list.

RETURNS

(nonzero value)—The pointer to the use item found.

NULL—No item of the type specified by *ulType* was found or *pCurrentItem* was not in the list.

HOW TO CALL

Call this method to enumerate items of a specific type in the in-use list. Do not use the pNext pointer in the USEITEM structure to traverse the in-use linked list.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpAddToObjUseList -145, wpDeleteFromObjUseList -148,
wpFindViewItem -152

NOTES

This method is extremely useful for obtaining information about which views are currently open for an object or which shadows of an object are currently awake.

SAMPLE CODE

The following sample shows how to use the wpFindUseItem method to get information about currently open views of an object. The function QueryOpenViews returns the number of open views for an object.

```
ULONG QueryOpenViews(WPObject *Object)
{
    PUSEITEM pCurrent;
    ULONG    ulCount = 0;

    /* use a for loop to find all the open views for this object */

    for (pCurrent = _wpFindUseItem(Object, USAGE_OPENVIEW, NULL);
        pCurrent;
        pCurrent = _wpFindUseItem(Object, USAGE_OPENVIEW, pCurrent) )
    {
        /* One was found. Increment the count. */

        ulCount++;
    }

    return(ulCount);
}
```

● **wpFindViewItem** **WPObject instance**
method (Warp only)

Finds a VIEWITEM in the in-use list of an object.

SYNTAX

PVIEWITEM wpFindViewItem (**ULONG** *flViews*, **PVIEWITEM**
pCurrentItem)

PARAMETERS

flViews - input

The types of views to search for. The following flags can be ORed together:

Define		Description
VIEW_CONTENTS	0x01	Icon view for folders.
VIEW_SETTINGS	0x02	Settings view.
VIEW_RUNNING	0x08	Program view for programs and program files.
VIEW_DETAILS	0x10	Details view for folders.
VIEW_TREE	0x20	Tree view for folders.
VIEW_ANY	0xFFFFFFFF	Any view.

pCurrentItem - input

The address of a VIEWITEM (pg.155) in the list after which to start the search. If NULL, the search will start with the first item matching *flViews*. Note: this value is a VIEWITEM not a USEITEM.

RETURNS

(nonzero value)—The pointer to the VIEWITEM found.

NULL—No VIEWITEM for the views specified in *flViews* was found.

HOW TO CALL

Call this method to find a VIEWITEM in the in-use list. wpFindUseItem can also be used to find VIEWITEM structures.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpFindUseItem - 150

In-Use List: Structures

LINKITEM**(4 bytes)**

LinkObj (WPObj *)

0

The WPSHADOW object this use item represents.

Used by: wpAddToObjUseList -145, wpDeleteFromObjUseList -148, wpFindUseItem -150

MEMORYITEM**(4 bytes)**

cbBuffer (ULONG)

0

The number of bytes in the memory block this use item represents. The memory directly follows this structure.

Used by: wpAddToObjUseList -145, wpDeleteFromObjUseList -148, wpFindUseItem -150

RECORDITEM**(12 bytes)**

hwndCnr (HWND)

0

The container the object has been inserted into.

pRecord (PMINIRECORDCORE)

4

The container record representing this object.

ulUser (ULONG)

8

For application use.

Used by: wpAddToObjUseList -145, wpDeleteFromObjUseList -148, wpFindUseItem -150

USEITEM**(8 bytes)****type** (ULONG) 0

The type of this use item.

Define	Description
USAGE_MEMORY	1 Usually added by wpAllocMem and is followed by a MEMORYITEM structure.
USAGE_RECORD	4 Added when the object is placed in a container and is followed by a RECORDITEM structure.
USAGE_OPENVIEW	5 Added when a view of an object is opened and is followed by a VIEWITEM structure.
USAGE_LINK	6 Added when a shadow of an object is created and is followed by a LINKITEM structure.
USAGE_OPENFILE	20 Added when a data file opens a program and is followed by a VIEWFILE structure.

pNext (PUSEITEM) 4

Pointer to the next use item. Use wpFindUseItem instead of referencing this pointer directly.

Used by: wpAddToObjUseList -145, wpDeleteFromObjUseList -148, wpFindUseItem -150**VIEWFILE****(16 bytes)****ulMenuId** (ULONG) 0

The menu ID if this is an association or an item from the menu page.

handle (LHANDLE) 4

The handle for the open view. This is the HAPP of the program started with WinStartApp.

hwndCnr (HWND) 8

Reserved for system use only.

pRecord (PMINIRECORDCORE) 12

Reserved for system use only.

Used by: wpAddToObjUseList -145, wpDeleteFromObjUseList -148, wpFindUseItem -150

VIEWITEM**(20 bytes)**

view (ULONG) 0

The identifier of the view this use item represents. See Appendix E for a list of predefined views.

handle (LHANDLE) 4

The handle for the open view. For OPEN_RUNNING, this is the HAPP of the program started with WinStartApp. For other predefined views, this is an hwnd.

ulViewState (ULONG) 8

View state flags:

Define	Description	
VIEWSTATE_OPENING	0x01	The view is being opened.
VIEWSTATE_OBJECTDELETED	0x02	The object is being deleted.
VIEWSTATE_USERHIDDEN	0x04	The view was hidden by the user.
VIEWSTATE_POPULATING	0x08	The view is populating.

hwndCnr (HWND) 12

Reserved for system use only.

pRecord (PMINIRECORDCORE) 16

Reserved for system use only.

Used by: wpAddToObjUseList -145, wpDeleteFromObjUseList -148, wpFindUseItem -150, wpFindViewItem -152



Views

A view allows the user to interact with an object. For example, the `WPFolder` class provides icon, tree, and details views to show the contents of a folder, and the `WPPalette` class provides the palette view to display the palette itself. Each object has a settings view that displays its properties and allows the user to alter them. Most available views are listed in the Open pulldown of the pop-up menu. (In Warp, the Settings view menu item was moved to the top level menu.) When the user selects a view to open from a pop-up menu or double-clicks on an object, the system calls the `wpViewObject` method.

Many Workplace Shell classes have predefined views listed in Appendix E, and application developers can define new views for their subclasses. Each view is represented by a numeric value, and applications must define their views using values greater than or equal to `OPEN_USER`. If you wish to change the behavior of a view already defined by your parent class or process one of your application-defined views, override `wpOpen`. All Workplace views, once opened, must be registered with the system and added to the object's in-use list with `wpRegisterView` and `wpAddToObjUseList` respectively. It is the responsibility of the programmer to call these two methods when an application-defined view is opened.

Registering a View with Workplace

When a Workplace Shell object opens a view, the `wpRegisterView` method should be called to notify the system to register the view. When a view is registered, it will be placed in the window list and subclassed by the shell to provide pop-up menu support from the system menu. The system will also change the title of the opened view window to include the name of the view.

The In-Use List

`wpAddToObjUseList` must be called to add a `USAGE_OPENVIEW` item to the in-use list whenever a view is created. This will put hash marks on the object's icon and ensure that the object will not go dormant while the view is open. This also enables the shell to switch to the view when the object is double-clicked on by the user or when `wpSwitchTo` is called. (See Chapter 6 for more information about in-use lists.)

The Default View

The default view is the view opened when the user double-clicks on the object, when the Open conditional cascade in the pop-up menu is selected without opening its pulldown, or when `wpOpen` is called with `OPEN_DEFAULT` as the *ulView* parameter. When the `WPObject` class processes the `wpModifyPopupMenu` method, the menu item for the default view is given a check mark in the Open pulldown of the pop-up menu. Users set the default view of a `WPFileSystem` descendant via the Menu page; applications can set and query this value for any object using `wpSetDefaultView` and `wpQueryDefaultView` respectively.

Restrictions and Warnings

- When the system places a check mark next to the default view in the Open pulldown of the pop-up menu, it assumes the menu item IDs for application-defined views are the same as their corresponding view constants.
- When defining a new view and adding it to the in-use list for an object, if you do not set the handle field in the `VIEWITEM` structure to be an `HWND`, you must override `wpSwitchTo` to provide the functionality for that view.

Instance Methods

wpClose closes all views of an object (pg. 158).

wpHide hides all views of an object (pg. 159).

wpOpen opens a view of an object (pg. 160).

wpQueryButtonAppearance returns the type of minimize button for an object's views (pg. 162).

wpQueryConcurrentView returns the view open behavior of an object (pg. 164).

wpQueryDefaultView returns the default view of an object (pg. 165).

wpQueryMinWindow returns the behavior of the minimize button for an object's views (pg. 166).

wpRegisterView registers a view with the system (pg. 167).

wpRestore restores all views of an object (pg. 168).

wpSetButtonAppearance sets the type of minimize button for an object's views (pg. 169).

wpSetConcurrentView sets the view open behavior of an object (pg. 170).

wpSetDefaultView sets the default view of an object (pg. 171).

wpSetMinWindow sets the behavior of the minimize button for an object's views (pg. 172).

wpSwitchTo switches to an open view of an object (pg. 173).

wpViewObject opens a view of an object (pg. 174).

wpWaitForClose (Warp) pauses a thread until the specified view is closed (pg. 176).

● **wpClose** **WPObject instance method**

Closes all views of an object.

SYNTAX

BOOL wpClose ()

PARAMETERS

none

RETURNS

TRUE—The views were successfully closed.

FALSE—An error occurred.

HOW TO CALL:

Call this method on any object to close all of its views.

HOW TO OVERRIDE

Overriding this method will not provide notification each time a view is closed by the user. This method is only called by the system when an object is closed as a result of a workarea folder closing or when an object is deleted.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpHide -159, wpOpen -160, wpRestore -168

● wpHide WPObject instance method

Hides all views of an object.

SYNTAX

BOOL wpHide ()

PARAMETERS

none

RETURNS

TRUE—The views were successfully hidden.

FALSE—An error occurred.

HOW TO CALL

Call this method on any object to hide all of its views.

HOW TO OVERRIDE

Overriding this method will not provide notification each time a view is hidden by the user. This method is only called by the system when an object is hidden as a result of a workarea folder hiding or minimizing.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpClose -158, wpOpen -160, wpRestore -168

● wpOpen WPObject instance method

Opens a view of an object.

SYNTAX

HWND wpOpen (**HWND** *hwndCnr*, **ULONG** *ulView*, **ULONG** *param*)

PARAMETERS

hwndCnr - input

The window handle for the container from which this object is opened. This can be set to NULLHANDLE. The system usually sets this parameter when the user opens an object.

ulView - input

The numeric value that represents the view to open. See Appendix E for a list of predefined Workplace Shell views. Application-defined views must be greater than or equal to OPEN_USER.

param - input

An application-defined parameter for the view. For predefined Workplace views, this value should be NULLHANDLE.

RETURNS

(nonzero value)—The handle for the opened view.

NULLHANDLE—An error occurred.

HOW TO CALL

Call this method on any object to open a view. In most cases, it is preferable to call wpViewObject. wpOpen will open the view regardless

of whether a view is already opened. However, `wpViewObject` will query the object's open behavior and first call `wpSwitchTo`. `wpViewObject` eventually calls `wpOpen`.

HOW TO OVERRIDE

Override this method to open an application-defined view or to change the behavior of an existing view. Whenever an object's view is opened, this method is called. Call the parent method to open any views not defined by your subclass.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpClose` -158, `wpclsQueryDefaultView` -178, `wpHide` -159, `wpQueryDefaultView` -165, `wpRegisterView` -167, `wpRestore` -168, `wpSetDefaultView` -171, `wpSwitchTo` -173, `wpViewObject` -174

NOTES

It is not necessary to check if a view of the same type is already opened from the override of `wpOpen`. When the user opens an object, the system calls `wpViewObject` which checks for open views and then calls `wpOpen`. Providing this functionality in the `wpOpen` method would be duplicate code.

SAMPLE CODE

The following sample demonstrates how to call `wpOpen` to open the Icon view of the System Setup folder.

```
SOMAny *Folder;

/* get the object pointer for the system setup folder */
Folder = _wpclsQueryFolder(_WPObject, "<WP_CONFIG>", TRUE);
if (Folder)
{
    /* open the folder and unlock the object because we are through
       with it */
    _wpOpen(Folder, NULLHANDLE, OPEN_CONTENTS, 0);
    _wpUnlockObject(Folder);
}
```


The default value for an object is DEFAULTBUTTON; therefore, the object will always use the class default unless wpSetButtonAppearance is called to change the value.

SAMPLE CODE

The following sample demonstrates how to use wpQueryButtonAppearance. It is a code snippet from a function called OpenView which could be called by the wpOpen override when the application-defined view is requested:

```
HWND OpenView(SOMAny *Object)
{
    ULONG ulButtonAppearance;
    ULONG ulFrameFlags = FCF_TITLEBAR | FCF_SYSMENU | FCF_SIZEBORDER
                        | FCF_ICON | FCF_SHELLPOSITION;
    HWND hwndFrame;
    /* The parameter, Object, passed into this function is the
     * pointer of the object passed into the wpOpen override.
     */
    ulButtonAppearance = _wpQueryButtonAppearance(Object);

    /* If this object uses the class default, query the default
     appearance for the class.
     */
    if (ulButtonAppearance == DEFAULTBUTTON)
        ulButtonAppearance =
            _wpClsQueryButtonAppearance(_somGetClass(Object));

    if (ulButtonAppearance == HIDEBUTTON)
        ulFrameFlags |= FCF_HIDEMAX;
    else
        ulFrameFlags |= FCF_MINMAX;
    /* Now you are ready to create the frame with the correct frame
     control flags.
     */

    ...
    return(hwndFrame);
}
```

● **wpQueryConcurrentView** **WPObj** instance method

Returns the view open behavior of an object.

SYNTAX

ULONG wpQueryConcurrentView()

PARAMETERS

none

RETURNS

Define		Description
CCVIEW_DEFAULT	0	Use the system default.
CCVIEW_ON	1	Open a new view each time it is requested by the user.
CCVIEW_OFF	2	Open a new view only if none yet exist.

HOW TO CALL

Call this method on any object to determine its view open behavior. The user chooses this setting on the Window page of the settings notebook. Call wpclsQuerySetting (Warp only) on the WPSystem class to determine the system default (see Appendix F).

HOW TO OVERRIDE

Override this method to temporarily change the response an object has to this method. Call wpSetConcurrentView to permanently change the object's view open behavior.

OTHER INFO

Include file: wobject.h



SEE ALSO

wpclsQuerySetting -126, wpOpen -160, wpSetConcurrentView -170, wpViewObject -174

NOTES

The `wpViewObject` method will call `wpQueryConcurrentView` to determine whether to try to switch to an existing view of the same type before calling `wpOpen`.

● **wpQueryDefaultView** **WPObj**ect instance method

Returns the default view of an object.

SYNTAX

ULONG wpQueryDefaultView()

PARAMETERS

none

RETURNS

The default view for this object. See Appendix E for a list of predefined views.

HOW TO CALL

Call this method on any object to determine its default view.

HOW TO OVERRIDE

Override this method to temporarily change the response an object has to this method. Call `wpSetDefaultView` to permanently change the default view for this object. Override `wpclsQueryDefaultView` to change the class default.

OTHER INFO

Include file: `wobject.h`

SEE ALSO

`wpclsQueryDefaultView` -178, `wpOpen` -160, `wpSetDefaultView` -171, `wpViewObject` -174

NOTES

The `wpViewObject` and `wpOpen` methods will call this method if `OPEN_DEFAULT` is the requested view.

● **wpQueryMinWindow** **WPObj** instance method

Returns the behavior of the minimize button for an object's views.

SYNTAX

ULONG wpQueryMinWindow()

PARAMETERS

none

RETURNS

The behavior of the minimize button for any view of this object:

Define		Description
MINWIN_DEFAULT	0	Use the system default behavior for minimized windows.
MINWIN_HIDDEN	1	Hide the window.
MINWIN_VIEWER	2	Minimize the window to the Minimized Window Viewer.
MINWIN_DESKTOP	3	Minimize the window to the Desktop.

HOW TO CALL

Call this method to query the behavior of the minimize button for an object's views. Call wpclsQuerySetting (Warp only) on the WPSys class to determine the system default behavior (see Appendix F).

HOW TO OVERRIDE

Override this method to temporarily change the response an object has to this method. Call wpSetMinWindow to permanently change the minimize button behavior for this object.

OTHER INFO

Include file: wplib.h

SEE ALSO

wpclsQuerySetting -126, wpQueryButtonAppearance -162,
wpSetMinWindow -172, wpSetButtonAppearance -169

NOTES

The system calls this method when the view of an object is minimized by the user. If the view has a hide button instead of a minimize button, the view will always hide and this method is not called.

● **wpRegisterView** **WPObj**ect instance method

Registers a view with the system.

SYNTAX

BOOL wpRegisterView(**HWND** *hwndFrame*, **PSZ** *pszViewTitle*)

PARAMETERS

hwndFrame - input

The frame window handle for this view.

pszViewTitle - input

The title for this view.

RETURNS

TRUE—The view was registered successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method on any object to register its view with the system. Applications that define new views should call this method when the view is opened.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wobject.h`

SEE ALSO

`wpOpen` -160

NOTES

When this method is called, the system will change the title of *hwndFrame* to be the object title followed by the view name specified in *pszViewTitle*.

RESTRICTIONS/WARNINGS

- A view that is registered does not need the FCF_TASKLIST frame control flag because Workplace will put the view in the window list.
- The window handle passed to `wpRegisterView` must be a frame window. Note that the window handle passed into a dialog window procedure is the frame, but the window handle passed into a standard window procedure is the client. To get the frame handle from the client, call `WinQueryWindow` on the client passing `QW_PARENT`.

● **wpRestore** **WPObject instance method**

Restores all views of an object.

SYNTAX

BOOL wpRestore ()

PARAMETERS

none

RETURNS

TRUE—The views were successfully restored.

FALSE—An error occurred.

HOW TO CALL

Call this method on any object to restore all of its views.

HOW TO OVERRIDE

Overriding this method will not provide notification each time a view is restored by the user. This method is only called by the system when an object is restored as a result of a workarea folder being restored.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

wpClose -158, wpHide -159, wpOpen -160, wpSwitchTo -173

NOTES

When a view is restored, the window is returned to its original size, before it was minimized or hidden.

● wpSetButtonAppearance WPObject instance method

Sets the type of minimize button for an object's views.

SYNTAX

VOID wpSetButtonAppearance(**ULONG** *ulButtonType*)

PARAMETERS

ulButtonType - input

The type of minimize button to use.

Define		Description
HIDEBUTTON	1	Hides the window when pressed by the user.
MINBUTTON	2	Minimizes the window when pressed by the user.
DEFAULTBUTTON	3	Uses the default returned from wpclsQueryButtonAppearance.

RETURNS

none

HOW TO CALL

Call this method on any object to set its minimize button appearance. If the hide button is used, the views will always hide when it is pressed. However, if a minimize button is used, the button will function according to the value returned from wpQueryMinWindow. The user chooses this setting on the Window page of the settings notebook.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpclsQueryButtonAppearance -177, wpQueryButtonAppearance -162, wpQueryMinWindow -166

NOTES

The Settings view of an object always has a hide button regardless of the object's button appearance setting.

●

wpSetConcurrentView

WPObject instance method

Sets the view open behavior of an object.

SYNTAX

VOID wpSetConcurrentView(**ULONG** *ulCCView*)

PARAMETERS

ulCCView - input

The concurrent view behavior:

Define		Description
CCVIEW_DEFAULT	0	Use the system default.
CCVIEW_ON	1	Open a new view each time it is requested by the user.
CCVIEW_OFF	2	Open a new view only if none yet exist.

RETURNS

none

HOWTO CALL

Call this method on any object to set its view open behavior. The user chooses this setting on the Window page of the settings notebook. Call wpclsSetSetting (Warp only) on the WPSystem class to set the system default behavior (see Appendix F).

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpclsSetSetting` -131, `wpQueryConcurrentView` -164

NOTES

The `wpViewObject` method will call `wpQueryConcurrentView` to determine whether to try to switch to an existing view of the same type before calling `wpOpen`.

● **wpSetDefaultView** **WPObj** instance method

Sets the default view of an object.

SYNTAX

BOOL wpSetDefaultView(**ULONG** *ulView*)

PARAMETERS

ulView - input

The new default view for this object. See Appendix E for a list of possible views. Do not set this parameter to `OPEN_DEFAULT`.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method on any object to change its default view.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

wpclsQueryDefaultView -178, wpOpen -160, wpQueryDefaultView -165, wpViewObject -174

NOTES

The wpViewObject and wpOpen methods will query this value with wpQueryDefaultView if OPEN_DEFAULT is the requested view.

● **wpSetMinWindow** **WPObjct instance method**

Sets the behavior of the minimize button for an object's views.

SYNTAX

VOID wpSetMinWindow(**ULONG** *ulMinWindow*)

PARAMETERS

ulMinWindow - input

The behavior of the minimize button for any view of this object:

Define		Description
MINWIN_DEFAULT	0	Use the system default behavior for minimized windows.
MINWIN_HIDDEN	1	Hide the window.
MINWIN_VIEWER	2	Minimize the window to the Minimized Window Viewer.
MINWIN_DESKTOP	3	Minimize the window to the Desktop.

RETURNS

none

HOW TO CALL

Call this method to set the behavior of the minimize button for an object's views. This does not apply to views with a hide button. Call wpclsSetSetting (Warp only) on the WPSysSystem class to set the system default behavior (see Appendix F).

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpclsSetSetting` -131, `wpQueryButtonAppearance` -162,
`wpQueryMinWindow` -166, `wpSetButtonAppearance` -169

NOTES

The system calls `wpQueryMinWindow` when the minimize button of a view is pressed by the user.

● **wpSwitchTo** **WPObj** instance method

Switches to an open view of an object.

SYNTAX

BOOL wpSwitchTo (**ULONG** *View*)

PARAMETERS

View - input

The numerical value that represents the view to switch to. See Appendix E for a list of Workplace Shell predefined views.

RETURNS

TRUE—A view of this type was successfully found and brought to the foreground.

FALSE—No views of this type were found or an error occurred.

HOW TO CALL

Call this method on any object to bring a previously opened view to the foreground. The system will search for the first `USAGE_VIEW` item in the in-use list corresponding to the specified view. If this method is called and the view is not yet opened, it will return **FALSE**. If an open view is desired, call `wpViewObject` instead. `wpViewObject` will first call `wpSwitchTo` to find an existing view and then call `wpOpen` if a new open view is needed.

HOW TO OVERRIDE

This method should only need to be overridden to process application-defined views that are added to the in-use list by setting the handle field in the VIEWITEM structure to something other than an HWND.

OTHER INFO

Include file: wpobject.h

SEE ALSO

wpClose -158, wpHide -159, wpOpen -160, wpRestore -168,
wpViewObject -174

● wpViewObject WPObject instance method

Opens a view of an object.

SYNTAX

HWND wpViewObject (**HWND** *hwndCnr*, **ULONG** *ulView*, **ULONG** *param*)

PARAMETERS

hwndCnr - input

The window handle for the container from which this object is opened. This can be set to NULLHANDLE. The system usually sets this parameter when the user opens an object.

ulView - input

The numerical value that represents the view to open. See Appendix E for a list of predefined Workplace views.

param - input

Application-defined parameter for the view. For predefined Workplace views, this value should be NULLHANDLE.

RETURNS

(nonzero value)—The handle for the opened view.

NULLHANDLE—An error occurred.

HOW TO CALL

Call this method on any object to open a view. Calling this method is preferred over calling wpOpen directly.

HOW TO OVERRIDE

Override this method to change its behavior, but do not use this override to process the opening of application-defined views; `wpOpen` should be overridden for that purpose.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpClose` -158, `wpclsQueryDefaultView` -178, `wpHide` -159, `wpOpen` -160, `wpQueryDefaultView` -165, `wpRegisterView` -167, `wpRestore` -168, `wpSetDefaultView` -171, `wpSwitchTo` -173

NOTES

The `WPObj` class processes this method first by checking the concurrent view behavior of the object with `wpQueryConcurrentView`. If concurrent views are allowed, `wpOpen` is called, passing the same parameters that were passed to `wpViewObject`. If concurrent views are not allowed, an existing view is searched for with `wpSwitchTo`. If `wpSwitchTo` then returns `FALSE`, `wpOpen` is called. The net effect is that a view of the requested type is brought to the foreground.

SAMPLE CODE

The following sample demonstrates how to call `wpViewObject` by opening the Icon view of the System Setup folder.

```
SOMAny *Folder;

/* get the object pointer for the system setup folder */
Folder = _wpclsQueryFolder(_WPObj, "<WP_CONFIG>", TRUE);
if (Folder)
{
    /* open the folder and unlock the object because we are through
       with it */
    _wpViewObject(Folder, NULLHANDLE, OPEN_CONTENTS, 0);
    _wpUnlockObject(Folder);
}
```

● **wpWaitForClose** **WPObj**ect instance
method (**Warp** only)

Pauses a thread until the specified view is closed.

SYNTAX

ULONG wpWaitForClose(**LHANDLE** *lhView*, **ULONG** *ulViews*, **LONG** *lTimeOut*, **BOOL** *bAutoClose*)

PARAMETERS

lhView - input

The window handle or HAPP of the view to wait for. If **NULLHANDLE** is specified, this method will use the *ulViews* parameter.

ulViews - input

The numerical value that represents the type of view to wait for. This method will wait until all views of this type have closed. See Appendix E for a list of predefined Workplace views. This parameter is used only if *lhView* is **NULLHANDLE**.

lTimeOut - input

The maximum time to wait before returning.

bAutoClose - input

When set to **TRUE**, the system will close the specified view(s), if still open.

RETURNS

0—The method completed successfully.

-1—An error occurred.

Otherwise—The return code from WinWaitEventSem.

HOW TO CALL

Call this method from a separate thread to block until a view of an object has closed.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.h`

Class Methods

wpclsQueryButtonAppearance returns the default type of the minimize button for views (pg. 177).

wpclsQueryDefaultView returns the default view for a class (pg. 178).

wpclsQueryObjectFromFrame (Warp) returns the pointer to the object that owns a specified view's frame window (pg. 179).

● **wpclsQueryButtonAppearance** **WPObj class method**

Returns the default type of minimize button for views.

SYNTAX

ULONG wpclsQueryButtonAppearance()

PARAMETERS

none

RETURNS

Define		Description
HIDEBUTTON	1	Use a hide button.
MINBUTTON	2	Use a minimize button.

HOW TO CALL

Call this method on any class object to determine the default button appearance of that class.

HOW TO OVERRIDE

Override this method to provide the class default button appearance. The WPObj class object will respond to this method by returning the value set on the Window page of the System object.

OTHER INFO

Include file: wproject.h

SEE ALSO

wpOpen -160, wpQueryButtonAppearance -162,
wpSetButtonAppearance -169

NOTES

The Settings view of an object always has a hide button regardless of the object's button appearance setting.

SAMPLE CODE

See the sample for wpQueryButtonAppearance which also shows when to use wpclsQueryButtonAppearance.

● **wpclsQueryDefaultView** **WPObject class method**

Returns the default view for a class.

SYNTAX

ULONG wpclsQueryDefaultView()

PARAMETERS

none

RETURNS

The default view for this class. See Appendix E for a list of possible views. Application-defined views above OPEN_USER can also be returned.

HOW TO CALL

Call this method on any class object to determine its default view.

HOW TO OVERRIDE

Override this method to provide the class default view. The parent method does not have to be called.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpOpen -160, wpQueryDefaultView -165, wpSetDefaultView -171,
wpViewObject -174

RESTRICTIONS/WARNINGS

- When an object is created, its default view is set to this value. The default view of the object is saved and the class method will not be called again to determine the class default. Therefore, if the class default view is changed by changing the return from `wpclsQueryDefaultView`, it will not affect objects that have already been created.

●	<code>wpclsQueryObjectFromFrame</code>	WPDesktop class method (Warp only)
---	---	---

Returns the pointer to the object that owns a specified view's frame window.

SYNTAX

WPObj * `wpclsQueryObjectFromFrame` (**HWND** *hwndFrame*)

PARAMETERS

hwndFrame - input

The handle of a top-level window.

RETURNS

(nonzero value)—The pointer to the object that owns this view.

NULL—No associated object could be found.

HOW TO CALL

Call this method on the Desktop class object, `_WPDesktop`, to find the object associated with a view.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpdesk.h`

SEE ALSO

`wpOpen` -160, `wpViewObject` -174

8

Settings Notebooks

All Workplace objects inherit the Settings view `OPEN_SETTINGS` from the `WPObject` class. `WPObject` opens this view by creating a notebook using the PM notebook control and calls `wpAddSettingsPages` on the object. Each class overrides `wpAddSettingsPages` to add its pages to the notebook. These pages are dialogs that enable the user to change the object's instance data. To implement a settings notebook page, first design a dialog resource using the dialog box editor that comes with the OS/2 toolkit and define the corresponding dialog procedure. Then override `wpAddSettingsPages` to call `wpInsertSettingsPage` and pass the required information.

All settings pages for an object are added to the notebook by `wpAddSettingsPage` each time the Settings view for the object is opened. However, the dialog resource for each page is not loaded by the system until the user first selects that page. Once a dialog for a page has been loaded, it remains as a child of that page until the notebook is closed.

All Workplace classes have a method for every page added to the settings notebook. If you do not want a specific page to appear in the

notebook, override the corresponding method listed in Appendix C and return `SETTINGS_PAGE_REMOVED`. These settings page methods can also be called directly from a `wpAddSettingsPages` override.

Restrictions and Warnings

- The Workplace classes generally implement the `wpAddSettingsPages` method by passing the `BKA_LAST` flag to `wpInsertSettingsPage`. This always adds the page to the bottom of the notebook.
- Verify that your settings page dialog procedure sets the focus on one of its controls whenever it returns `TRUE` from `WM_INITDLG`. If it returns `FALSE`, the focus will automatically be set for you. Failing to ensure that focus is on one of the controls could cause random desktop windows to receive the focus as settings pages are turned by the user.

Instance Methods

`wpAddSettingsPages` adds pages to an object's settings notebook (pg. 181).

`wpInsertSettingsPage` inserts an individual page into the settings notebook (pg. 183).

● **`wpAddSettingsPages`** **WPObj** instance method

Adds pages to an object's settings notebook.

SYNTAX

BOOL `wpAddSettingsPages`(**HWND** *hwndNotebook*)

PARAMETERS

hwndNotebook - input

The handle of the settings notebook created by the `WPObj` class.

RETURNS

TRUE—The settings pages were successfully added.

FALSE—An error occurred.

HOW TO CALL

This method is generally only called by the system. If an application calls this method, the `wpInsertSettingsPage` method must be overridden and its functionality must be completely replaced.

HOW TO OVERRIDE

Override this method to add pages to the Settings view. Call the parent class at any time to allow ancestor classes to add their settings pages as well. The system will not open the settings notebook if FALSE is returned from this method.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpInsertSettingsPage` -183

SAMPLE CODE

The following sample demonstrates how to override `wpAddSettingsPages` to add pages to the settings notebook.

```
SOM_Scope BOOL  SOMLINK mcls_wpAddSettingsPages(MyClass *somSelf,
                                                HWND hwndNotebook)
{
    PAGEINFO pageinfo;
    BOOL fReturn;
    /* This is a help subtable in memory. The first value is the size
     * in words of each entry. Next are pairs of dialog controls with
     * their corresponding help panel ids. The last entry MUST be 0,
     * 0. A help subtable is only required if you want to implement
     * context sensitive help.
     */
    HELPSUBTABLE HelpSubTable [] = { 2,
                                      IDC_ENTRY, IDHELP_ENTRY,
                                      IDC_UNDO,  IDHELP_UNDO,
                                      0,        0        };

    fReturn = parent_wpAddSettingsPages(somSelf, hwndNotebook);
```

```

/* add another settings page here... */

memset((PCH)&pageinfo,0,sizeof(PAGEINFO));
pageinfo.cb = sizeof(PAGEINFO);
pageinfo.hwndPage = NULLHANDLE; // use the resource
pageinfo.usPageStyleFlags = BKA_MAJOR; // major tab
pageinfo.usPageInsertFlags = BKA_FIRST; // add to top of
// notebook

pageinfo.pfnwp = MyPageDlgProc; // window proc
pageinfo.resid = hmod; // .DLL module handle
pageinfo.dlgid = IDD_MYPAGE; // ID of dialog
pageinfo.pszName = "~My Page"; // Tab text
pageinfo.pCreateParams = somSelf; // pass object ptr to
// the dialog

pageinfo.idDefaultHelpPanel = RES_MYPAGE; // resource id of
// general help

pageinfo.pszHelpLibraryName = "MYCLASS.HLP"; // help library name
pageinfo.pHelpSubtable = HelpSubTable; // help subtable

if (! _wpInsertSettingsPage( somSelf, hwndNotebook, &pageinfo ))
    fReturn = FALSE;
return fReturn;
}

```

wpInsertSettingsPage **WPObj** instance method

Inserts an individual page into the settings notebook.

SYNTAX

ULONG wpInsertSettingsPage(**HWND** *hwndNotebook*, **PAGEINFO** *ppageinfo*)

PARAMETERS

hwndNotebook - input

The handle of the settings notebook created by the WPObj class. Use the *hwndNotebook* parameter passed into the wpAddSettingsPages override when calling this method.

ppageinfo - input

A pointer to a PAGEINFO structure (pg. 187) containing information about the page to add.

RETURNS

(nonzero value)—The ID of the page inserted.
0—An error occurred.

HOW TO CALL

This method is only called from an override of `wpAddSettingsPages`, and it should be called once for each page that is to be added to the settings notebook.

HOW TO OVERRIDE

This method is generally not overridden unless the entire functionality of the Settings view is being replaced.

OTHER INFO

Include files: `wobject.h` and `pmwp.h` define:
`INCL_WINWORKPLACE`

SEE ALSO

`wpAddSettingsPages` -181, `wpclsQuerySettingsPageSize` -185

NOTES

If a dialog resource ID is passed in the `PAGEINFO` structure, the system will load the dialog when the user turns to that page.

The system makes a copy of the `PAGEINFO` structure passed into *ppageinfo*. Therefore, *ppageinfo* can be freed after the call to this method is made.

SAMPLE CODE

See the sample for `wpAddSettingsPages` for an example of how to use this method.

Class Methods

`wpclsQuerySettingsPageSize` returns the size of the settings pages for a class (pg. 185).

`wpclsSetSettingsPageSize (Warp)` sets the size of the settings pages for a class (pg. 186) .

● wpclsQuerySettingsPageSize WPObjct class method

Returns the size of the settings notebook pages for a class.

SYNTAX

BOOL wpclsQuerySettingsPageSize(**PSIZEL** *pSizl*)

PARAMETERS

pSizl - input/output

A pointer to a **SIZEL** structure (pg. 189) to be filled with the desired size, in dialog units, for the settings notebook pages for this class.

RETURNS

TRUE—The call was successful.

FALSE—An error occurred.

HOW TO CALL

This method can be called to query the default settings page size for a class. Most applications should have no need to call this method.

HOW TO OVERRIDE

Override this method to return the default settings page size for your class. The parent method does not need to be called but is useful in determining the maximum size required. The **WPObjct** class processes this method by returning the system default size for settings pages.

OTHER INFO

Include files: `wpobject.h` and `pmgpi.h`

SEE ALSO

`wpAddSettingsPages` -181, `wpclsSetSettingsPageSize` -186,
`wpInsertSettingsPage` -183

NOTES

Override this method if the dialogs for your settings pages are unusually long or wide. This method is only called once for each object, when the settings notebook is first opened. The object then saves its settings notebook size in the user ini file. Therefore, if the class default changes, it will not affect objects that have already had their Settings view opened.

SAMPLE CODE

The following sample is an override of `wpclsQuerySettingsPageSize`. The largest dialog in this class is 300×200 dialog units.

```
SOM_Scope BOOL  SOMLINK mclsM_wpclsQuerySettingsPageSize
                                     (M_MyClass*somSelf,
                                     PSIZEL pSizl)
{
    /* call the parent method to determine its size requirements */
    parent_wpclsQuerySettingsPageSize(somSelf, pSizl);

    /* Set the height and width to the values for this class or the
     * values returned from the parent method, whichever is larger.
     */
    pSizl->cx = max(300, pSizl->cx);
    pSizl->cy = max(200, pSizl->cy);

    return (TRUE);
}
```

● **wpclsSetSettingsPageSize** **WObject class** **method (Warp only)**

Sets the size of the settings notebook pages for a class.

SYNTAX

BOOL wpclsSetSettingsPageSize(**PSIZEL** *pSizl*)

PARAMETERS

pSizl - input

A pointer to a **SIZEL** structure (pg. 189) with the desired size, in dialog units, for the settings notebook pages for this class.

RETURNS

TRUE—The call was successful.

FALSE—An error occurred.

HOWTO CALL

This method can be called to set the default settings page size for a class. It is preferable to override `wpclsQuerySettingsPageSize` to specify this value.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wobject.h` and `pmgpi.h`

SEE ALSO

`wpAddSettingsPages` -181, `wpclsQuerySettingsPageSize` -185,
`wpInsertSettingsPage` -183

Settings Notebooks: Structures

PAGEINFO	(52 bytes)
cb (ULONG) The size of the structure, in bytes, including cb itself.	0
hwndPage (HWND) The window handle of the page to add if it has already been created. This can be set to NULLHANDLE, and a dialog resource ID can be passed in the dlgid field instead.	4
pfnwp (PFNWP) The dialog procedure for the page if dlgid was specified.	8
resid (ULONG) The module handle for the .DLL that contains the dialog resource if dlgid is specified.	12
pCreateParams (PVOID) The creation parameters passed into WinLoadDlg when the system loads the dialog. This is only valid if dlgid is specified.	16
dlgid (USHORT) The resource ID of the dialog to load into the page when it is first selected by the user. If this field is nonzero, pfnwp and resid must also be specified.	20
usPageStyleFlags (USHORT) Page style flags for the notebook control:	22

Define		Description
BKA_STATUSTEXTON	0x001	Turn on status area text. This is not needed if SETTINGS_PAGE_NUMBERS is specified in usSettingsFlags.
BKA_MAJOR	0x040	This page has a major tab.
BKA_MINOR	0x080	This page has a minor tab.

usPageInsertFlags (USHORT) 24

Page insertion order constants for the notebook control:

Define		Description
BKA_LAST	0x02	Insert as the last page.
BKA_FIRST	0x04	Insert as the first page.
BKA_NEXT	0x08	Insert as the page after ulPageInsertId.
BKA_PREV	0x10	Insert as the page before ulPageInsertId.

usSettingsFlags (USHORT) 26

Settings notebook flags:

Define		Description
SETTINGS_PAGE_NUMBERS	0x01	When this flag is set, the system will place page numbers on the status line of the notebook in the format: "Page n of n."

pszName (PSZ) 28

The title for the tab on this page. Place a ~ in front of a letter to make it the mnemonic. Be careful not to duplicate mnemonics of tabs for an object's settings notebook.

idDefaultHelpPanel (USHORT) 32

The extended help panel for the dialog. Workplace will create a help instance for the notebook and associate this resource ID with the dialog window. This can be 0 if no help is available.

usReserved2 (USHORT) 34

Reserved for system use.

pszHelpLibraryName (PSZ) 36

The name of the help library file that contains idDefaultHelpPanel and the IDs in pHlpSubtable. This can be NULL if no help is available.

pHelpSubtable (PUSHORT)

40

A PHELPSUBTABLE pointer (defined in PMHELP.H). This subtable will be added to the help table for the settings notebook. This can be 0 if there is no help available or if only the extended help panel will be used. The following is an example of a help subtable in memory:

```
HELPSUBTABLE HelpSubTable [] = { 2,
                                   ID_DLGCONTROL1, IDHELP_DLGCONTROL1,
                                   ID_DLGCONTROL2, IDHELP_DLGCONTROL2,
                                   0,                0                };
```

where ID_DLGCONTROLx are dialog control IDs and IDHELP_DLGCONTROLx are help panel IDs.

hmodHelpSubtable (HMODULE)

44

This value is not used by the system. Set it to 0.

ulPageInsertId (ULONG)

48

The ID of the page to place this page before or after. This value is only used if BKA_NEXT or BKA_PREV are specified in usPageInsertFlags.

Used by: wpInsertSettingsPage -183

SIZE**(8 bytes)****cx** (ULONG)

0

Width

cy (ULONG)

4

Height

Used by: wpclsQuerySettingsPageSize -185, wpclsSetSettingsPageSize -186

●9

Pop-Up Menus

When a user summons a pop-up menu on an object, Workplace creates a standard menu and then calls `wpFilterPopupMenu` and `wpModifyPopupMenu` on the object. These methods allow the object to filter out items that are unwanted and add new items. The filtered items are not actually removed from the menu until after both methods are called. If the user presses F1 while the pop-up menu is up, the `wpMenuItemHelpSelected` method is called. This is covered in the Help chapter (pg. 339).

Restrictions and Warnings

- Not all menu items have `CTXT_` constants defined for removal with `wpFilterPopupMenu`. However, each item has a `WPMENUID_` constant defined in `wpobject.h` which can be removed with the `MM_DELETEITEM` message from a `wpModifyPopupMenu` override.

Instance Methods

wpDisplayMenu (Warp) creates and displays the pop-up menu for an object (pg. 191).

wpFilterPopupMenu filters unwanted standard pop-up menu items (pg. 193).

wpInsertPopupMenuItems inserts items into a pop-up menu (pg. 195).

wpMenuItemSelected notifies an object when the user has selected an item from its pop-up menu (pg. 198).

wpModifyPopupMenu allows the object to add items to or alter the pop-up menu (pg. 199).

● **wpDisplayMenu** **WPObj**ect instance method (Warp only)

Creates and displays the pop-up menu for an object.

SYNTAX

```
HWND wpDisplayMenu(HWND hwndOwner, HWND hwndClient,
                   POINTL ptlPopupPt, ULONG ulMenuType,
                   ULONG ulReserved)
```

PARAMETERS

hwndOwner - input

The window handle of the owner for this menu. This parameter is required.

hwndClient - input

The window handle of the window on which this menu is displayed. This window must be owned by *hwndOwner*. *hwndClient* will be given the focus before the menu is displayed. If set to `NULLHANDLE`, *hwndOwner* will be given the focus instead.

ptlPopupPt - input

The coordinates at which to display the menu. These coordinates are relative to *hwndClient*, if specified, and *hwndOwner* otherwise. If *ulMenuType* is `MENU_OBJECTPOPUP` and *hwndClient* is an icon or tree view of a container, this parameter is ignored and the menu will be displayed next to the object's icon. If *ulMenuType* is `MENU_TITLEBARPULLDOWN`, this parameter is ignored and the menu is displayed at the position of the system menu.

ulMenuType - input

The type of menu to display:

Define	Description
<code>MENU_OBJECTPOPUP</code>	1 A pop-up for an object displayed in a view (that is, an icon in a folder view).
<code>MENU_OPENVIEWPOPUP</code>	2 A pop-up for an open view.
<code>MENU_TITLEBARPULLDOWN</code>	3 A title bar pulldown menu.

ulReserved - reserved

Set this parameter to zero.

RETURNS

(nonzero value)—The window handle of the menu.

`NULLHANDLE`—An error occurred.

HOWTO CALL

Call this method at any time to display a pop-up menu.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wobject.h`

SEE ALSO

`wpFilterPopupMenu` -193, `wpModifyPopupMenu` -199

● wpFilterPopupMenu WPObject instance method

Filters out unwanted standard pop-up menu items.

SYNTAX

ULONG wpFilterPopupMenu(**ULONG** *ulFlags*, **HWND** *hwndCnr*,
BOOL *fMultiSelect*)

PARAMETERS

ulFlags - input

Multiple context menu items ORed together. This parameter contains the flags for desired pop-up menu items. The following is a list of pop-up menu item flags.

Define	Menu Item
CTXT_CRANOTHER	0x001 Create another
CTXT_OPEN	0x002 Open
CTXT_WINDOW	0x004 Window
CTXT_CLOSE	0x008 Close
CTXT_SETTINGS	0x010 Settings
CTXT_PRINT	0x020 Print
CTXT_HELP	0x040 Help
CTXT_DELETE	0x080 Delete
CTXT_COPY	0x100 Copy
CTXT_MOVE	0x200 Move
CTXT_SHADOW	0x400 Create shadow
CTXT_PROGRAM	0x800 Program
CTXT_ICON	0x1000 Icon view
CTXT_TREE	0x2000 Tree view
CTXT_DETAILS	0x4000 Details view
CTXT_FIND	0x8000 Find
CTXT_SELECT	0x10000 Select
CTXT_ARRANGE	0x20000 Arrange

Define	Menu Item
CTXT_SORT	0x40000 Sort
CTXT_SHUTDOWN	0x80000 Shut down
CTXT_LOCKUP	0x100000 Lockup
CTXT_PALETTE	0x200000 Palette
CTXT_REFRESH	0x400000 Refresh
CTXT_PICKUP	0x800000 Pickup (Warp)
CTXT_PUTDOWN	0x1000000 Putdown (Warp)
CTXT_PUTDOWNCANCEL	0x2000000 Cancel drag

hwndCnr - input

The window handle of the container of the object where the pop-up menu was summoned. This value is sometimes NULLHANDLE.

fMultiSelect - input

If set to TRUE, this pop-up menu is for multiple objects at once.

RETURNS

Valid context menu flags for this object ORed together.

HOWTO CALL

This method is generally only called by the system.

HOWTO OVERRIDE

Override this method to remove unwanted pop-up menu items. The system will pass all the CTXT_ flags in *ulFlags*, and it is up to the classes to remove items and return the result. Call the parent method first and store the return code; then remove any unwanted items and return the result.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpInsertPopupMenuItems` -195, `wpMenuItemSelected` -198,
`wpModifyPopupMenu` -199, `wpModifyStyle` -108

NOTES

Some object styles will cause WPObject to remove pop-up menu items in its implementation of wpFilterPopupMenu. See wpModifyStyle for more information on object styles.

SAMPLE CODE

The following samples demonstrates how to override wpFilterPopupMenu.

```
SOM_Scope ULONG  SOMLINK mcls_wpFilterPopupMenu(MyClass *somSelf,
                                                ULONG ulFlags,
                                                HWND hwndCnr,
                                                BOOL fMultiSelect)
{
    ULONG ulReturn;

    /* get the parent's pop-up menu flags */
    ulReturn = parent_wpFilterPopupMenu(somSelf, ulFlags, hwndCnr, fMultiSelect);

    /* remove the settings menu item */
    ulReturn &= ~CTXT_SETTINGS;

    return(ulReturn);
}
```

●	wpInsertPopupMenuItems	WPObject instance method
---	-------------------------------	-------------------------------------

Inserts menu items into a pop-up menu.

SYNTAX

BOOL wpInsertPopupMenuItems(**HWND** *hwndMenu*, **ULONG** *iPosition*,
HMODULE *hmod*, **ULONG** *MenuID*,
ULONG *SubMenuID*)

PARAMETERS

hwndMenu - input

The window handle of the pop-up menu in which to insert the items.

iPosition - input

The position to place the items. This parameter has no effect in OS/2 2.1 but works properly in Warp.

hmod - input

The handle for the module that contains the menu items to add.

MenuID - input

The ID of the menu resource that contains the new menu items to load and add to *hwndMenu*.

SubMenuID - input

The ID of the submenu of *hwndMenu* in which the new menu items are to be inserted.

Define		Submenu Item
WPMENUID_PRIMARY	0	Add the items to the top-level pop-up menu.
WPMENUID_OPEN	1	Open
WPMENUID_HELP	2	Help
WPMENUID_PRINT	3	Print
WPMENUID_SELECT	4	Select
WPMENUID_SORT	5	Sort

RETURNS

TRUE—The items were successfully added.

FALSE—An error occurred.

HOW TO CALL

This method should be called from an override of `wpModifyPopupMenu` to add items to the pop-up menu. Override `wpMenuItemSelected` for notification when the user selects one of the added items.

There is no restriction on the value of *MenuID*, but the menu items contained in the menu represented by *MenuID* must be above `WPMENUID_USER`.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

wpFilterPopupMenu -193, wpMenuItemSelected -198,
wpModifyPopupMenu -199

NOTES

Be careful not to duplicate mnemonics already in use by the standard menu items.

SAMPLE CODE

The following sample demonstrates how to call wpInsertPopupMenuItems.

From the .h file:

```
#define ID_POPUPMENU 4000
#define IDM_ITEMONE WPMENUID_USER
#define IDM_ITEMTWO WPMENUID_USER+1
```

From the .rc file:

```
/* This entire submenu is not added to the pop-up. Only the items,
 * IDM_ITEMONE and IDM_ITEMTWO, will be added.
 */
MENU ID_POPUPMENU LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    MENUITEM "~Item one", IDM_ITEMONE
    MENUITEM "I~tem two", IDM_ITEMTWO
END
```

From the .c file:

```
SOM_Scope BOOL SOMLINK mcls_wpModifyPopupMenu(MyClass *somSelf,
                                                HWND hwndMenu,
                                                HWND hwndCnr,
                                                ULONG iPosition)
{
    /* call the parent method first */
    parent_wpModifyPopupMenu(somSelf, hwndMenu, hwndCnr, iPosition);

    /* add our items to the primary menu */
    _wpInsertPopupMenuItems(somSelf, hwndMenu, iPosition,
                           hmod, ID_POPUPMENU, WPMENUID_PRIMARY);

    return(TRUE);
}
```

● **wpMenuItemSelected** **WPObj** instance method

Notifies an object when the user has selected an item from its pop-up menu.

SYNTAX

BOOL wpMenuItemSelected(**HWND** *hwndFrame*, **ULONG** *ulMenuId*)

PARAMETERS

hwndFrame - input

The frame window handle of the view containing the object.

ulMenuId - input

The ID of the menu item selected by the user.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method is generally only called by the system when the user selects a pop-up menu item.

HOW TO OVERRIDE

Override this method to add functionality for a menu item. This should be done if your class added new items to the pop-up menu, or if you wish to replace the behavior of an existing item. All menu items in Warp are defined with **WPMENUID_*** constants in `wpobject.h`. If the item is not processed, call the parent method.

OTHER INFO

Include file: `wpobject.h`

SEE ALSO

`wpFilterPopupMenu` -193, `wpInsertPopupMenuItems` -195,
`wpModifyPopupMenu` -199

● wpModifyPopupMenu WPObject instance method

Allows the object to add items to or alter the pop-up menu.

SYNTAX

BOOL wpModifyPopupMenu(**HWND** *hwndMenu*, **HWND** *hwndCnr*,
ULONG *iPosition*)

PARAMETERS

hwndMenu - input

The window handle of the pop-up menu to modify.

hwndCnr - input

The window handle of the container of the object where the pop-up menu was summoned. This value is sometimes NULLHANDLE.

iPosition - input

The position to place the menu.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method is generally only called by the system when the user summons a pop-up menu.

HOW TO OVERRIDE

Override this method to modify the pop-up menu. Change the menu by calling wpInsertPopupMenuItems and then call the parent method. Menu control messages can then be sent to *hwndMenu* to further modify the menu. Items can be removed using the menu control messages and the WPMENUID_* constants defined in wobject.h.

OTHER INFO

Include file: wobject.h

SEE ALSO

wpFilterPopupMenu -193, wpInsertPopupMenuItems -195,
wpMenuItemSelected -198

●10

Drag/Drop

Direct manipulation, or drag/drop, is a feature of Presentation Manager. Workplace Shell makes extensive use of this interface to allow users to manipulate objects easily. This chapter assumes knowledge of the drag/drop programming interface.

Dragging from Workplace Shell

Every folder in Workplace is a frame window with a container control as its client. When the container sends the frame the messages to start direct manipulation, Workplace starts the drag using different combinations of the following rendering mechanisms and rendering formats:

"<DRM_OBJECT, DRF_OBJECT>"	Specified for every object dragged from Workplace.
"<DRM_OS2FILE, DRF_UNKNOWN>"	Specified for nontext files (such as folders).
"(DRM_OS2FILE, DRM_PRINT) X (DRF_TEXT)"	Specified for printable text files.

File objects can be dragged from Workplace to any PM application that supports DRM_PRINT or DRM_OS2FILE. Any window running on the Workplace process can accept a drop when the rendering mechanism is DRM_OBJECT, DRM_PRINT, or DRM_OS2FILE. When the target has completed processing, the DM_ENDCONVERSATION message should be sent to pDragItem->hwndItem. If the rendering mechanism is DRM_OBJECT, pDragItem->ullItemID indicates the container record for the object dragged. The macro OBJECT_FROM_PREC can be used to extract the SOM object pointer from the container record.

Dragging to Workplace Shell

If an application initiates a drag of a file and the file is dragged to a folder, the rendering mechanism should be DRM_OS2FILE. If pDragItem->hstrContainerName or pDragItem->hstrSourceName is set, Workplace will perform the action (copy or move). If pDragItem->hstrContainerName is NULL, Workplace will send a DM_RENDERPREPARE message to the source window followed by a DM_RENDER message, and then the source must do the operation.

If the file is dropped on the shredder, DRM_OS2FILE is specified, and pDragItem->hstrContainerName or pDragItem->hstrSourceName is set, then Workplace will perform the deletion of the file. If neither string is set, or DRM_DISCARD is specified, then the shredder will send a DM_DISCARD message to the source window.

If the file is dropped on a printer, DRM_OS2FILE is specified, and pDragItem->hstrContainerName or pDragItem->hstrSourceName is set, then Workplace will print the file. If neither string is set, or DRM_PRINT is used, then the printer sends a DM_PRINTOBJECT message to the source window.

Workplace Drag Methods

Workplace provides methods to notify objects when direct manipulation is occurring. Many of the direct manipulation messages map to methods that can be overridden by any object to support drag/drop.

wpDraggedOverObject notifies an object that it is being dragged over another object (pg. 202).

wpDragOver notifies an object that it is the current target in a drag operation (pg. 204).

wpDrop notifies the target object that a drop has occurred (pg. 206).

wpDroppedOnObject notifies an object that it has been dropped on another object (pg. 209).

wpEndConversation is documented as a notification for the source object that the target sent a DM_ENDCONVERSATION message. Currently, this method is never called. See wpFormatDragItem for more information.

wpFormatDragItem allows the source object in a drag operation to format its drag information (pg. 209).

wpRender is documented as a notification for the source object if a DM_RENDER message is sent by the target. Currently, this method is never called. See wpFormatDragItem for more information.

wpRenderComplete is documented as a notification for the target object if a DM_RENDERCOMPLETE message is sent by the source. Currently, this method is never called.

● **wpDraggedOverObject** **WPObj** instance method

Notifies an object that it is being dragged over another object.

SYNTAX

MRESULT wpDraggedOverObject(**WPObj** **DraggedOverObject*)

PARAMETERS

DraggedOverObject - input

The target over which this object is currently being dragged.

RETURNS

The return from this method is the same as the return from the DM_DRAGOVER message:

Short1: Drop indicator:

Define	Description
DOR_NODROP	0 The object cannot be dropped at this time. The target may change state at some point which would allow the drop to occur.
DOR_DROP	1 The object can be dropped. Default operation must be set to indicate the default operation.
DOR_NEVERDROP	3 The object can never be dropped.

Short2: Default operation:

Define	Description
DO_COPY	0x10 The copy operation.
DO_MOVE	0x20 The move operation.
DO_LINK	0x18 The link operation.
DO_CREATE	0x40 The create operation.
DO_UNKNOWN	0xBFF Unknown operation.

HOW TO CALL

This method can be called from an override of wpDragOver to allow the source object to decide if a drag operation is allowed. The system only calls this method on a file system object when it is dragged over a program object.

HOW TO OVERRIDE

Override this method to specify that a drag operation is allowed when the target asks for the source to decide.

OTHER INFO

Include file: wobject.h and pmstdlg.h

SEE ALSO

wpDragOver -204, wpDrop -206, wpDroppedOnObject -209

● **wpDragOver** **WPObject instance method**

Notifies an object that it is the current target in a drag operation.

SYNTAX

MRESULT wpDragOver(**HWND** *hwndCnr*, **PDRAGINFO** *pdrgInfo*)

PARAMETERS

hwndCnr - input

The handle to the target container in the drag operation.

pdrgInfo - input

A pointer to the DRAGINFO structure (pg. 211) passed from the DM_DRAGOVER message.

RETURNS

The return from this method is the same as the return from the DM_DRAGOVER message:

Short1: Drop indicator:

Define	Description
DOR_NODROP	0 The object cannot be dropped at this time. The target may change state at some point when the drop is allowed.
DOR_DROP	1 The object can be dropped. Default operation must be set to indicate the default operation.
DOR_NODROPOP	2 The object cannot be dropped at this time because the current operation is not acceptable.
DOR_NEVERDROP	3 The object can never be dropped.

Short2: Default operation:

Define	Description
DO_COPY	0x10 The copy operation.
DO_MOVE	0x20 The move operation.
DO_LINK	0x18 The link operation.
DO_CREATE	0x40 The create operation.
DO_UNKNOWN	0xBFF Unknown operation.

HOW TO CALL

This method is generally called by the system when the container receives the DM_DRAGOVER message. wpDragOver is called on the current target object for the drag operation.

HOW TO OVERRIDE

Override this method to specify whether this drag operation is allowed.

OTHER INFO

Include file: wpobject.h and pmstdlg.h

SEE ALSO

wpDraggedOverObject -202, wpDrop -206, wpDroppedOnObject -209

SAMPLE CODE

The following sample demonstrates how to override wpDragOver. In this example, only WPDataFile descendants will be allowed to be dropped on objects of this class.

```
SOM_Scope MRESULT SOMLINK mcls_wpDragOver(MyClass *somSelf,
                                           HWND hwndCnr,
                                           PDRAINFORM pdrgInfo)
{
    PDRAINFORM pDragItem;
    ULONG ulCount;
    USHORT usDrop = DOR_DROP, usDefaultOp = DO_MOVE;
    SOMAny *Object;

    /* Loop through the drag item structures to see what is being
     * dragged over this object.
     */
    for (ulCount=0; usDrop == DOR_DROP && ulCount < pdrgInfo->cditem;
         ulCount++)
    {
        pDragItem = DrgQueryDragitemPtr(pdrgInfo, ulCount);
        /*
         * For each drag item, verify it is a WPS object
         * "<DRM_OBJECT,DRF_OBJECT>". If it is, only allow data files to
         * be dropped.
         */
        if (DrgVerifyRMF(pDragItem, "DRM_OBJECT", "DRF_OBJECT"))
```

```

{
    /* pDragItem->ulItemID is the RECORDCORE structure for the
     * object. OBJECT_FROM_PREC is a macro defined in wpobject.h
     * that will give the pointer of a Workplace object given its
     * container record.
     */
    Object = OBJECT_FROM_PREC(pDragItem->ulItemID);

    /* If it is not a data file, do not allow the drop. (wpdataf.h
     * must be included to use the _WPDataFile variable)
     */
    if (!_somIsA(Object, _WPDataFile) )
        usDrop = DOR_NEVERDROP;
}
else
    usDrop = DOR_NEVERDROP;
}

return MRFROM2SHORT(usDrop, usDefaultOp);
}

```

● **wpDrop** **WPObject instance method**

Notifies the target object that a drop has occurred.

SYNTAX

MRESULT wpDrop(**HWND** *hwndCnr*, **PDRAGINFO** *pdrgInfo*,
 PDRAGITEM *pdrgItem*)

PARAMETERS

hwndCnr - input

The handle to the target container in the drag operation.

pdrgInfo - input

The pointer to the DRAGINFO structure (pg. 211) passed from the DM_DROP message.

pdrgItem - input

A pointer to a DRAGITEM structure (pg. 212). On the first call of wpDrop, *pdrgItem* points to the first DRAGITEM of *pdrgInfo*. If RC_DROP_ITEMCOMPLETE is returned, wpDrop will be called for each item of *pdrgInfo*, and *pdrgItem* will point to its corresponding DRAGITEM. See the sample code below.

RETURNS

Define	Description
RC_DROP_ITEMCOMPLETE	1 Only the item specified by <i>pdrgItem</i> has been processed. <i>wpDrop</i> should be called again for each item.
RC_DROP_DROPCOMPLETE	2 The entire drop has been successfully processed.
RC_DROP_RENDERING	0 The target cannot render the object and source rendering is requested. Workplace will send required messages to the source.
RC_DROP_ERROR	-1 An error has occurred.

HOW TO CALL

This method is generally called only by the system on the current target object when the container receives the *DM_DROP* message.

HOW TO OVERRIDE

Override this method to process a drop on this object. Override *wpDragOver* to prevent unwanted items from being dropped. The *WPFolder* class will process this override by performing copy, move, create shadow, or create from template operations depending on the value of *pdrgItem->usOperation*. When files are dropped on *WPProgram* objects, they process *wpDrop* by running the program specified in its *PROGDETAILS* and passing the file as a parameter. The *WPShrepper* class will delete objects or files from the *wpDrop* override. Call the parent method only if you wish to let the parent perform these functions.

OTHER INFO

Include file: *wpobject.h* and *pmstdlg.h*

SEE ALSO

wpDraggedOverObject -202, *wpDragOver* -204, *wpDroppedOnObject* -209

SAMPLE CODE

The following sample demonstrates how to override wpDrop. This override is from the same class as the wpDragOver sample. Only objects descendant from WPDataFile were allowed to be dropped on objects of this class. Since wpDragOver filtered out unwanted drag operations, it is guaranteed that at the point when the wpDrop method is called, all the items are data file objects.

```
SOM_Scope MRESULT SOMLINK mcls_wpDrop(MyClass *somSelf,
                                         HWND hwndCnr,
                                         PDRAGINFO pdrgInfo,
                                         PDRAITEM pdrgItem)
{
    SOMAny *Object;

    /* Process this drop one item at a time using the pdrgItem
     * pointer. If we wanted to, we could have looped through the drag
     * items from the pdrgInfo structure and processed all the items
     * in this call.
     */

    /* ulItemID is the container record for the object.
     * OBJECT_FROM_PREC returns the object pointer given the record
     * pointer.
     */
    Object = OBJECT_FROM_PREC(pdrgItem->ulItemID);

    /* Perform an action on this object. In this example we'll do
     * something simple and set the readonly attribute on the object.
     */
    _wpSetAttr(Object, _wpQueryAttr(Object) | FILE_READONLY);

    /* Return RC_DROP_ITEMCOMPLETE so wpDrop will be called once for
     * every item in this drag operation.
     */
    return (MRESULT) RC_DROP_ITEMCOMPLETE;
}
```

● wpDroppedOnObject WPObjec instance method

Notifies an object that it has been dropped on another object.

SYNTAX

MRESULT wpDroppedOnObject(**WPObjec** **DroppedOnObject*)

PARAMETERS

DroppedOnObject - input

The target on which this object has just been dropped.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called during the override of wpDrop to allow the source object to process the drop operation. Call this method only when wpDraggedOverObject is also called on the source object to decide if a drag operation is allowed (this is done from the wpDragOver override). The system only calls this method on a file system object when it is dropped on a program object.

HOW TO OVERRIDE

Override this method to process a drop operation when the target requests it. This is analogous to source rendering.

OTHER INFO

Include file: wproject.h

SEE ALSO

wpDraggedOverObject -202, wpDragOver -204, wpDrop -206

● wpFormatDragItem WPObjec instance method

Allows the source object in a drag operation to format its drag information.

SYNTAX

BOOL wpFormatDragItem(**PDRAGITEM** *pdrgItem*)

PARAMETERS

pdrgItem - input

A pointer to the DRAGITEM (pg. 212) structure that represents this object.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method is generally only called by the system to notify the object to initialize its DRAGITEM structure.

HOW TO OVERRIDE

Override this method to change the DRAGITEM structure. You can call the parent method first to allow the parent classes and WPObject to first set up *pdrgItem* and then make the desired changes.

OTHER INFO

Include file: wobject.h and pmstdlg.h

SEE ALSO

wpDraggedOverObject -202, wpDragOver -204, wpDrop -206,
wpDroppedOnObject -209

NOTES

If source rendering is desired, override wpFormatDragItem. Change the rendering mechanism, if desired, and free the hstrSourceName in the DRAGITEM structure.

Currently, the methods wpEndConversation and wpRender do not function as they are documented. When the target sends the DM_RENDERPREPARE, DM_RENDER, and DM_ENDCONVERSATION messages, they are sent to the window handle stored in *pdrgItem*->hwndItem. This window does not process the messages to call the notification methods. To work around this problem, create an invisible window to process these messages and change the *pdrgItem*->hwndItem to point to that window.

Drag/Drop: Structures

DRAGINFO

(20 bytes)

cbDraginfo (ULONG) 0
The size of this structure plus all the drag items.

cbDragitem (USHORT) 4
The size of DRAGITEM.

usOperation (USHORT) 6
The current drag operation:

Define	Description
DO_COPY	0x10 The copy operation.
DO_MOVE	0x20 The move operation.
DO_LINK	0x18 The link operation.
DO_CREATE	0x40 The create operation.
DO_UNKNOWN	0xBFF Unknown operation.

hwndSource (HWND) 8
The handle of the source window.

xDrop (SHORT) 12
The x coordinate for the drop position.

yDrop (SHORT) 14
The y coordinate for the drop position.

cditem (USHORT) 16
The number of drag items.

usReserved (USHORT) 18
Reserved value.

Used by: wpDragOver -204, wpDrop -206

DRAGITEM**(36 bytes)**

hwndItem (HWND)	0
The window to communicate with for this item.	
ulItemID (ULONG)	4
The ID of the item dragged. For Workplace objects, this is a pointer to the container record for the object. OBJECT_FROM_PREC can be used to obtain an object pointer from a container record.	
hstrType (HSTR)	8
The type of the item.	
hstrRMF (HSTR)	12
The rendering mechanism and format.	
hstrContainerName (HSTR)	16
The name of the source container.	
hstrSourceName (HSTR)	20
The name of the item at the source.	
hstrTargetName (HSTR)	24
The suggested name for the item at the target.	
cxOffset (SHORT)	28
The x offset of the image from the hotspot of the mouse pointer.	
cyOffset (SHORT)	30
The y offset of the image from the hotspot of the mouse pointer.	
fsControl (USHORT)	32
The source object control flags.	
fsSupportedOps (USHORT)	34
The operations supported by the source.	

Define	Description	
DO_COPYABLE	0x01	The item can be copied.
DO_MOVEABLE	0x02	The item can be moved.
DO_LINKABLE	0x04	The item can be linked.
DO_CREATEABLE	0x08	The item can be created.

Used by: wpDrop -206, wpFormatDragItem -209

●11

File System Objects

File system objects are all descendants from the base class WPFileSystem. Directories are descended from WPFolder, and files from WPDataFile. One advantage of defining a subclass of WPFileSystem is that its instances can be moved onto a diskette or across a LAN. This is because the object's class data is stored in the Extended Attributes (EAs) of the file or directory it represents.

Data Files

When a `WPDataFile` descendant saves its instance data, the class name of the object is written to the EAs. The class of a data file object is always determined by the class name specified in the `.CLASSINFO` EA, if present. Note that all file system classes can override `wpclsQueryInstanceType` and/or `wpclsQueryInstanceFilter` to specify lists of types or extensions of its instances. Workplace builds a list containing classes and their corresponding `.TYPE` and extension lists by calling `wpclsQueryInstanceType` and `wpclsQueryInstanceFilter` on each file system subclass. When a data file without a `.CLASSINFO` EA is awakened, the `.TYPE` and extension is checked against this list of classes. If no matching classes are found, the `WPDataFile` class is used.

Restrictions and Warnings

- The format of the `.CLASSINFO` EA is private and should not be manipulated directly. Only public methods can be used to change instance data defined by file system classes.
- File system objects are designed to have their file names correspond to their titles as much as possible. Therefore, when you call `wpSetTitle`, the file name is also changed.

Instance Methods

`wpQueryAttr` returns the file attributes of an object (pg. 215).

`wpQueryCreation` returns the creation date and time of an object (pg. 216).

`wpQueryEASize` returns the size of the extended attributes of an object (pg. 217).

`wpQueryFileName (Warp)` returns the physical file name of an object. `wpQueryRealName` provides the same functionality, but checks the size of the buffer. Therefore, `wpQueryRealName` should be used instead.

`wpQueryFileSize` returns the size of the file represented by an object (pg. 218).

wpQueryLastAccess returns the last access date and time of an object (pg. 219).

wpQueryLastWrite returns the last write date and time of an object (pg. 219).

wpQueryRealName returns the physical file name of an object (pg. 220).

wpQueryType returns the value of the .TYPE EA of an object (pg. 221).

wpRefresh refreshes the file system information of an object (pg. 222).

wpSetAttr sets the file attributes of an object (pg. 223).

wpSetRealName is documented as setting the physical file name of an object. However, this method only sets the WPFileSystem internal instance data to a new name and does not affect the underlying file name. It is, therefore, not useful.

wpSetType sets the .TYPE EA of an object (pg. 224).

wpVerifyUpdateAccess (Warp) verifies write access to a file system object (pg. 225).

● **wpQueryAttr** **WPFileSystem instance method**

Returns the file attributes of an object.

SYNTAX

ULONG wpQueryAttr()

PARAMETERS

none

RETURNS

The attribute flags of the file. Any combination of the following may be returned:

Define		Description
FILE_NORMAL	0x000	The file can be read from or written to. No attributes are specified.
FILE_READONLY	0x001	The file can be read from but not written to or deleted.
FILE_HIDDEN	0x002	The file is hidden and does not show up in normal directory listings. Hidden file objects are usually invisible in folders unless the Include page of the folder specifies otherwise.
FILE_SYSTEM	0x004	The file is a system file.
FILE_DIRECTORY	0x010	The file is a subdirectory.
FILE_ARCHIVED	0x020	The file has been archived.

HOW TO CALL

This method can be called at any time to determine the attributes of the file represented by a file system object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfsys.h and bsedos.h define:
INCL_DOSFILEMGR

SEE ALSO

wpSetAttr-223

● **wpQueryCreation** **WPFileSystem** instance method

Returns the creation date and time of an object.

SYNTAX

ULONG wpQueryCreation(FDATE *fdate, FTIME *ftime)

PARAMETERS

fdate - input/output

A pointer to an FDATE structure (pg. 228) which, upon return, will be filled with the creation date of the file represented by this object.

ftime - input/output

A pointer to an FTIME structure (pg. 228) which, upon return, will be filled with the creation time of the file represented by this object.

RETURNS

Reserved

HOW TO CALL

This method can be called at any time on any file system object. Note that the creation date and time is not supported by the FAT file system.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfsys.h and bsedos.h define:
INCL_DOSFILEMGR

SEE ALSO

wpQueryLastAccess-219, wpQuery LastWrite-219

● wpQueryEASize WPFileSystem instance method

Returns the size of the extended attributes of an object.

SYNTAX

ULONG wpQueryEASize()

PARAMETERS

none

RETURNS

The size, in bytes, of the extended attributes for the file represented by this object.

HOW TO CALL

This method can be called at any time on any file system object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpfsys.h`

SEE ALSO

`wpQueryFileSize` -218

● wpQueryFileSize WPFileSystem instance method

Returns the size of the file represented by an object.

SYNTAX

ULONG `wpQueryFileSize()`

PARAMETERS

none

RETURNS

The size of the file, in bytes, represented by this object.

HOW TO CALL

This method can be called at any time on any file system object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpfsys.h`

SEE ALSO

`wpQueryEASize` -217

● wpQueryLastAccess WPFileSystem instance method

Returns the last access date and time of an object.

SYNTAX

ULONG wpQueryLastAccess(**FDATE** *fdate, **FTIME** *ftime)

PARAMETERS

fdate - input/output

A pointer to an FDATE structure (pg. 228) which, upon return, will be filled with the last access date of the file represented by this object.

ftime - input/output

A pointer to an FTIME structure (pg. 228) which, upon return, will be filled with the last access time of the file represented by this object.

RETURNS

Reserved

HOW TO CALL

This method can be called at any time on any file system object. Note that the last access date and time is not supported by the FAT file system.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfsys.h and bsedos.h

define:

INCL_DOSFILEMGR

SEE ALSO

wpQueryCreation -216, wpQueryLastWrite -219

● wpQueryLastWrite WPFileSystem instance method

Returns the last write date and time of an object.

SYNTAX

ULONG wpQueryLastWrite(**FDATE** *fdate, **FTIME** *ftime)

PARAMETERS

fdate - input/output

A pointer to an FDATE structure (pg. 228) which, upon return, will be filled with the last write date of the file represented by this object.

ftime - input/output

A pointer to an FTIME structure (pg. 228) which, upon return, will be filled with the last write time of the file represented by this object.

RETURNS

Reserved

HOWTO CALL

This method can be called at any time on any file system object.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfsys.h and bsedos.h define:
INCL_DOSFILEMGR

SEE ALSO

wpQueryCreation -216, wpQueryLastAccess -219

● **wpQueryRealName WPFileSystem instance method**

Returns the physical file name of an object.

SYNTAX

BOOL wpQueryRealName(**PSZ** *pszFileName*, **PULONG** *pcb*, **BOOL** *bQualified*)

PARAMETERS

pszFileName - input/output

A pointer to the string buffer in which to return the real file name of the object. If set to NULL, the required size is returned in *pcb*.

pcb - input/output

The size of the *pszFileName* buffer. If *pszFileName* is set to NULL, the required size is returned.

bQualified - input

TRUE—Return the fully qualified path for the object.

FALSE—Only return the name of the file and not the full path.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time on any file system object. The physical file name and the title of a file system object are usually similar; however, the physical file name is modified to conform to any naming rules of the file system.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfsys.h

SEE ALSO

wpQueryTitle -118

● **wpQueryType** **WPFileSystem instance method**

Returns the value of the .TYPE EA of an object.

SYNTAX

PSZ wpQueryType()

PARAMETERS

none

RETURNS

The pointer to the string containing all the types in the .TYPE EA of the object. Multiple types are separated by the line feed character: '\n'.

HOW TO CALL

This method can be called at any time on a file system object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpfsys.h`

SEE ALSO

`wpSetType` -224

● wpRefresh WPFileSystem instance method

Refreshes the file system information of an object.

SYNTAX

BOOL wpRefresh(**ULONG** *ulView* **PVOID** *pReserved*)

PARAMETERS

ulView - input

If refreshing a folder view, set *ulView* to either `OPEN_CONTENTS`, `OPEN_TREE`, or `OPEN_DETAILS`. Otherwise, set it to zero.

pReserved - reserved

Set this value to zero.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time on any file system object to refresh its file system information. The `WPFolder` override of this method will refresh any open views to display the current contents of a folder. Workplace is notified by the file system when a file changes and updates the object by calling this method from a background thread. This method should be used to force an object to immediately reflect file system changes.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfsys.h

● wpSetAttr WPFileSystem instance method

Sets the file attributes of an object.

SYNTAX

BOOL wpSetAttr(**ULONG** *attrFile*)

PARAMETERS

attrFile - input

The new file attributes for the object. OR together any combination of attribute flags.

Define		Description
FILE_NORMAL	0x000	The file can be read from or written to. No attributes are specified.
FILE_READONLY	0x001	The file can be read from but not written to or deleted.
FILE_HIDDEN	0x002	The file is hidden and does not show up in normal directory listings. Hidden file objects are usually invisible in folders unless the Include page specifies otherwise.
FILE_SYSTEM	0x004	The file is a system file.
FILE_DIRECTORY	0x010	The file is a subdirectory.
FILE_ARCHIVED	0x020	The file has been archived.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOWTO CALL

This method can be called at any time to set the attributes of the file represented by a file system object. wpQueryAttr should be called first to determine the current attribute flags. Then make the desired modifications on the return code and pass it to wpSetAttr.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfsys.h and bsedos.h define:
INCL_DOSFILEMGR

SEE ALSO

wpQueryAttr -215

SAMPLE CODE

The following sample demonstrates how to call wpSetAttr. Object is a SOMAny pointer that has already been set to point to a file system object.

```
/* Make this object read only. */
_wpSetAttr(Object, _wpQueryAttr(Object) | FILE_READONLY);
```

● **wpSetType WPFileSystem instance method**

Sets the value of the .TYPE EA of an object.

SYNTAX

BOOL wpSetType(*PSZ pszTypes*, **PFEA2LIST** *pfeal*)

PARAMETERS

pszTypes - input

The pointer to the string containing the new value for the .TYPE EA of the object. Multiple types are separated by the line feed character: '\n'.

pfeal - reserved

Set this to zero.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOWTO CALL

This method can be called at any time on any file system object to set its .TYPE EA. Be aware that all of the types are stored in the same string, so when wpSetType is called, the *pszTypes* list replaces the previous list for the object. Call wpQueryType to determine the current

list of types. Discard the unwanted ones from the return code and add new ones if desired and pass the new string to `wpSetType`.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpfsys.h`

SEE ALSO

`wpQueryType` -221

●	<code>wpVerifyUpdateAccess</code>	<code>WPFileSystem</code> instance method (Warp only)
---	--	--

Verifies write access to a file system object.

SYNTAX

ULONG `wpVerifyUpdateAccess()`

PARAMETERS

none

RETURNS

NO_ERROR—Write access to this object is successful.
Otherwise—The error returned from `DosOpen`.

HOW TO CALL

This method can be called at any time on any file system object to verify that it can be updated.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpfsys.h`

Class Methods

wpclsQueryInstanceType returns the list of types for data files of this class (pg. 226)

wpclsQueryInstanceFilter returns the list of extensions for data files of this class (pg. 227)

● **wpclsQueryInstanceType** **WPFileSystem class method**

Returns the list of types for data files of this class.

SYNTAX

PSZ wpclsQueryInstanceType()

PARAMETERS

none

RETURNS

The pointer to the string containing the names of the file types supported by this class. Multiple types are separated by commas.

HOW TO CALL

This method is only called by the system when a class is loaded.

HOW TO OVERRIDE

Override this method to specify the types of data files that should use this class. Workplace will store this list and, whenever a data file is awakened, if the type of the file matches one in the list for this class, the data file will be awakened as an instance of this class.

OTHER INFO

Include files: wpfsys.h

SEE ALSO

wpclsQueryInstanceFilter -227, wpQueryType -221, wpSetType -224

● wpclsQueryInstanceFilter WPFileSystem class method

Returns the list of extensions for data files of this class.

SYNTAX

PSZ wpclsQueryInstanceFilter()

PARAMETERS

none

RETURNS

The pointer to the string containing the names of the file extensions supported by this class. Multiple extensions are separated by commas ("*.TXT,*.DOC").

HOW TO CALL

This method is only called by the system when a class is loaded.

HOW TO OVERRIDE

Override this method to specify the extensions of data files that should use this class. Workplace will store this list and, whenever a data file is awakened, if the extension of the file matches a filter in the list for this class, the data file will be awakened as an instance of this class.

OTHER INFO

Include files: wpfsys.h

SEE ALSO

wpclsQueryInstanceType -226

File System Objects: Structures

FDATE**(2 bytes)****day** (UINT)

(5 bits)

The day of the year.

month (UINT)

(4 bits)

The numeric representation for the month.

year (UINT)

(7 bits)

The number of years since 1980.

Used by: wpQueryCreation -216, wpQueryLastAccess -219, wpQueryLastWrite -219**FTIME****(2 bytes)****twosecs** (UINT)

(5 bits)

The number of two-second increments.

minutes (UINT)

(6 bits)

The number of minutes.

hours (UINT)

(5 bits)

The number of hours.

Used by: wpQueryCreation -216, wpQueryLastAccess -219, wpQueryLastWrite -219

●12

Folder Objects

Objects that represent file system directories are descendants of the WPFolder class. Folders are used to contain other objects. Folders automatically contain any file that is in the directory represented by the folder, but can also contain abstract or transient objects. This means that not all folder contents can be viewed from the file system. Some are stored in the OS2.INI file (abstracts) and some exist only in memory (transients). When the contents of a folder are loaded into memory, it is called populating the folder. This can be accomplished directly by calling the wpPopulate method. The contents of a folder can be enumerated with the wpQueryContent method. The content list of a folder contains only the objects that are currently awake. Be sure to populate a folder before querying its content list.

Workplace is designed such that all objects must be created in a folder. Once an object has been created, its container record can be inserted into any container using the wpCnrInsertObject method.

Restrictions and Warnings

- The container view functionality of a folder is tightly integrated with the `WPFolder` class. It is therefore impossible to create a class that is not descendant from `WPFolder` that can contain other objects. Folder objects must be real file system directories.

Instance Methods

wpAddFirstChild (Warp) awakens the first subfolder of a folder (pg. 231).

wpAddToContent (Warp) adds an object to a folder's content list (pg. 232).

wpCnrInsertObject inserts an object into the specified container (pg. 233).

wpCnrRemoveObject removes an object from the specified container (pg. 234).

wpContainsFolders returns whether a folder contains subfolders (pg. 235).

wpDeleteContents deletes the contents of a folder (pg. 236).

wpDeleteFromContent (Warp) removes an object from a folder's content list (pg. 237).

wpIsCurrentDesktop returns whether a desktop object is the current desktop (pg. 238).

wpIsDetailsColumnVisible (Warp) returns whether a details data column is visible (pg. 239).

wpIsSortAttribAvailable (Warp) returns whether a sort attribute is available (pg. 240).

wpModifyFldrFlags (Warp) sets the flags describing the current state of a folder (pg. 241).

wpPopulate awakens the contents of a folder (pg. 242).

wpQueryContent enumerates the contents of a folder (pg. 243).

wpQueryFldrAttr returns the attributes of a folder's view (pg. 245).

wpQueryFldrDetailsClass returns the class used to define the columns of the details view of a folder (pg. 247).

wpQueryFldrFlags returns the flags describing the current state of a folder (pg. 248).

wpQueryFldrFont returns the font used for the text in a folder view (pg. 248).

wpQueryFldrSortClass (Warp) returns the sort class for a folder (pg. 250).

wpQueryFolder (Warp) returns the folder that contains an object (pg. 251).

wpQueryNextIconPos returns the next icon position for an object being inserted into the icon view of a folder (pg. 251).

wpSetDetailsColumnVisibility (Warp) hides or unhides a details data column (pg. 252).

wpSetFldrAttr sets the attributes of a folder's view (pg. 253).

wpSetFldrDetailsClass sets the class used to define the columns of the details view of a folder (pg. 254).

wpSetFldrFlags sets the flags describing the current state of a folder (pg. 255).

wpSetFldrFont sets the font used for icon text in a folder view (pg. 256).

wpSetFldrSortClass (Warp) sets the sort class for a folder (pg. 257).

wpSetNextIconPos sets the next icon position for an object being inserted into a folder. Currently, this method has no effect.

wpSetSortAttribAvailable (Warp) adds or removes a sort attribute for a folder's Sort pulldown in the pop-up menu (pg. 258).

● **wpAddFirstChild** **WPFolder instance method**
 (Warp only)

Awakens the first subfolder of a folder.

SYNTAX

WPObject * wpAddFirstChild()

PARAMETERS

none

RETURNS

(nonzero value)—The object pointer of a child folder of the folder.
NULL—An error occurred.

HOW TO CALL

This method can be called at any time to awaken one child folder in a folder. The system calls this method on each visible folder in a tree view to show the proper expansion emphasis.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpPopulate -242

●	wpAddToContent	WPFolder instance method (Warp only)
---	-----------------------	---

Adds an object to a folder's content list.

SYNTAX

BOOL wpAddToContent (**WPObject** * *Object*)

PARAMETERS

Object - input

The pointer to the object to add.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method is called only by the system to notify a folder that an object must be added to its content list in memory.

HOW TO OVERRIDE

Override this method for notification when an object is added. This is useful for subclasses that define their own views. You must call the parent method.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpDeleteFromContent -237, wpPopulate -242, wpQueryContent -243

● wpCnrInsertObject WPObject instance method

Inserts an object into the specified container.

SYNTAX

BOOL wpCnrInsertObject(**HWND** *hwndCnr*, **PPOINTL** *pptlIcon*,
 PMINIRECORDCORE *preccParent*, **RECORDINSERT** *pRecInsert*)

PARAMETERS

hwndCnr - input

The window handle of the container control in which to insert the object. The container must be created from a thread running on the Workplace Shell process.

pptlIcon - input

A pointer to a POINTL structure (pg. 137) which contains the icon position at which to insert the object. This pointer must be set, but the POINTL structure can contain zeros if the view for this container does not support icon positions or the next available position should be used.

preccParent - input

A pointer to the parent record for this object. If *hwndCnr* is not a tree view or this record does not have a parent, set this to NULL.

pRecInsert - input

A pointer to a RECORDINSERT structure (pg. 267) which can be passed to specify more information about how this object is to be inserted into the container. This parameter can be set to NULL.

RETURNS

TRUE—The object was successfully inserted into the container.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time to insert an object into a container. Workplace will subclass the container and add context menu

and drag/drop support. These records must be removed with `wpCnrRemoveObject` or `wpclsRemoveObjects` when they are no longer needed.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpobject.h` and `pmstdlg.h` define:
`INCL_WINSTDCNR`

SEE ALSO

`wpclsInsertMultipleObjects` -264, `wpclsRemoveObjects` -265,
`wpCnrRemoveObject` -234

NOTES

To manipulate objects that have been added to a container with `wpCnrInsertObject`, use the normal `CM_*` messages for containers to obtain pointers to the `MINIRECORDCORE` structures. Workplace defines two useful macros in `wpobject.h`. Pass a container record for an object to `OBJECT_FROM_PREC` and it will return the object pointer. Pass a container record to `USERWORD_FROM_PREC` and it will return a pointer to a user-defineable `ULONG` for that record.

Objects must first be created in a folder and then can be inserted into other containers using `wpCnrInsertObject`. The Nowhere folder `<WP_NOWHERE>` can be used as a place to create objects when it does not make sense for them to exist in a folder.

● **`wpCnrRemoveObject` `WPObj` instance method**

Removes an object from the specified container.

SYNTAX

BOOL `wpCnrRemoveObject`(**HWND** *hwndCnr*)

PARAMETERS

hwndCnr - input

The window handle of the container control from which to remove the object.

RETURNS

TRUE—The object was successfully removed from the container.
FALSE—An error occurred.

HOWTO CALL

This method can be called at any time to remove an object from a container. This method can only be called on objects that have been inserted into *hwndCnr* with *wpCnrInsertObject* or *wpclsInsertMultipleObjects*.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO:

Include files: `wobject.h`

SEE ALSO:

wpclsInsertMultipleObjects - 264, *wpclsRemoveObjects* - 265,
wpCnrInsertObject - 233

● wpContainsFolders WPFolder instance method

Returns whether a folder contains subfolders.

SYNTAX

BOOL wpContainsFolders(**BOOL** **pfSubFolders*)

PARAMETERS

pfSubFolders - input/output

A pointer to a **BOOL** which, upon return, will be set to **TRUE** if the folder contains subfolders and **FALSE** if it does not.

RETURNS

TRUE—The method completed successfully.
FALSE—An error occurred.

HOWTO CALL

This method can be called at any time on any folder object to determine if it contains subfolders. The folder does not have to be populated.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO:

Include files: wpfolder.h

SEE ALSO

wpPopulate -242

● wpDeleteContents WPFolder instance method

Deletes the contents of a folder.

SYNTAX

BOOL wpDeleteContents(**ULONG** *fConfirmations*)

PARAMETERS

fConfirmations - input

OR together desired delete confirmation flags. If **CONFIRM_DELETE** is specified, the user will be prompted for each object deleted. If **CONFIRM_DELETEFOLDER** is specified, the user will be prompted if this object contains a folder. Call **wpQueryConfirmations** on any object to obtain the user confirmation preferences.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time on any folder object to delete its contents. **wpDeleteContents** will first call **wpPopulate**, and then call **wpDelete** on each object contained in the folder using the specified confirmation flags.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpConfirmDelete -40, wpDelete -51, wpPopulate -242,
wpQueryConfirmations -111

NOTES

wpDelete calls wpDeleteContents if the object being deleted is a folder; wpDeleteContents in turn calls wpDelete on each object in the folder. Therefore, wpDeleteContents will be called recursively for each subfolder in this folder tree.

RESTRICTIONS/WARNINGS:

- When calling this method on a folder that contains many objects, create a separate thread to make the call. This will avoid hanging the system while the deletions take place.
- wpDeleteContents calls wpPopulate first and then deletes the contents of the folder. Therefore, do not invoke this method on the self pointer passed to a wpPopulate override.

● wpDeleteFromContent	WPFolder instance method (Warp only)
------------------------------	---

Removes an object from a folder's content list.

SYNTAX

BOOL wpDeleteFromContent (**WPObject** * *Object*)

PARAMETERS

Object - input

The pointer to the object to be removed.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method is called only by the system to notify a folder that an object must be removed from its content list in memory.

HOW TO OVERRIDE

Override this method for notification when an object is removed. This is useful for subclasses that define their own views. You must call the parent method.

OTHER INFO

Include files: `wpfolder.h`

SEE ALSO

`wpAddToContent` -232, `wpPopulate` -242, `wpQueryContent` -243

● wpIsCurrentDesktop WPDesktop instance method

Returns whether a desktop object is the current desktop.

SYNTAX

BOOL `wpIsCurrentDesktop()`

PARAMETERS

none

RETURNS

TRUE—This object is the current desktop.

FALSE—This object is not the current desktop or an error occurred.

HOW TO CALL

This method can be called at any time on any desktop object to query if it is the currently active desktop. The active desktop is the one that opens when Workplace is loaded.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpdesk.h`

SEE ALSO

`wpclsQueryActiveDesktop` -259, `wpclsQueryActiveDesktopHWND` -260, `wpclsQueryObjectFromPath` -76

NOTES

It is possible for users to have multiple desktops on their machines but only one may be active at a time. If you have a pointer for a `WPDesktop` object, you can find out if it is the active desktop by calling `wpIsCurrentDesktop`. Note that since this is a `WPDesktop` method, it cannot be called on all folder instance pointers. It must be called on an instance pointer of `WPDesktop` or a subclass of `WPDesktop`. Another way to obtain the object pointer for the current desktop is to call `wpclsQueryObjectFromPath` passing "<WP_DESKTOP>" as the path.

●	<code>wpIsDetailsColumnVisible</code>	WPFolder instance method (Warp only)
---	--	---

Returns whether a details data column is visible.

SYNTAX

BOOL `wpIsDetailsColumnVisible` (**ULONG** *index*)

PARAMETERS

index - input

The index of the details data column.

RETURNS

TRUE—The column is visible.

FALSE—The column is hidden.

HOW TO CALL

Call the method at any time to determine if a column in a folder's details view is visible. The folder's details view columns are determined by the folder's details class. This class can be queried with `wpQueryFldrDetailsClass`.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpfolder.h`

SEE ALSO

wpclsQueryDetailsInfo -123, wpQueryFldrDetailsClass -247,
wpSetDetailsColumnVisibility -252

●	wpIsSortAttribAvailable	WPFolder instance method (Warp only)
---	--------------------------------	---

Returns whether a sort attribute is available.

SYNTAX

BOOL wpIsSortAttribAvailable (**ULONG** *index*)

PARAMETERS

index - input

The index of the details data column.

RETURNS

TRUE—The attribute is displayed in the Sort pulldown of the folder's pop-up menu.

FALSE—The attribute is not available.

HOW TO CALL

Call this method at any time to determine which sort attributes are available to the user in the pop-up menu of a folder. These sort attributes are those defined by the folder sort class. This class can be queried with wpQueryFldrSortClass.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpclsQueryDetailsInfo -123, wpIsDetailsColumnVisible -239,
wpQueryFldrSortClass -250, wpSetSortAttribAvailable -258

● **wpModifyFldrFlags** **WPFolder instance method**
(Warp only)

Sets the flags describing the current state of a folder.

SYNTAX

BOOL wpModifyFldrFlags(**ULONG** *ulFlags*, **ULONG** *ulFlagMask*)

PARAMETERS:

ulFlags - input

The folder flags to modify. OR together multiple flags to turn on or off more than one flag at once. The following are the documented folder flags:

Folder Flags

FOI_POPULATEDWITHALL
 FOI_POPULATEDWITHFOLDERS
 FOI_WORKAREA
 FOI_CHANGEFONT
 FOI_WAMINIMIZE
 FOI_WASTARTONRESTORE
 FOI_NOREFRESHVIEWS
 FOI_ASYNCREFRESHONOPEN
 FOI_TREEMPOPULATE
 FOI_REFRESHINPROGRESS
 FOI_FIRSTPOPULATE
 FOI_WAMCRINPROGRESS
 FOI_CNRBKGNDOLDFORMAT
 FOI_CHANGEICONBGNDCOLOR

ulFlagMask - input

The mask of flags indicating which will be set and which will be removed. For example, if *ulFlags* is FOI_WORKAREA | FOI_POPULATEDWITHALL and *ulFlagMask* is FOI_POPULATEDWITHALL,

FOI_WORKAREA will be removed and FOI_POPULATEDWITHALL will be set.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method on any folder to set its current folder flags. The FOI_WORKAREA flag is persistent when the folder goes dormant. The populate flags are not. Calling this method is preferable to using wpSetFldrFlags.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpQueryFldrFlags -248, wpSetFldrFlags -255

● **wpPopulate** **WPFolder instance method**

Awakens the contents of a folder.

SYNTAX

BOOL wpPopulate(**ULONG** *ulReserved*, **PSZ** *pszPath*, **BOOL** *fFoldersOnly*)

PARAMETERS

ulReserved - reserved

Set this to zero.

pszPath - input

The fully qualified path of the folder being populated. Call wpQueryRealName on the folder object to obtain this string.

fFoldersOnly - input

If TRUE, the folder will only be populated with objects descendent from WPFolder. If FALSE, all objects will be awakened.

RETURNS

TRUE—The call completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time on any folder object to populate its contents. If successful, `wpPopulate` will set the `FOI_POPULATEDWITHFOLDERS` flag and, if `fFoldersOnly` is set to `FALSE`, the `FOI_POPULATEDWITHALL` flag. If the folder is already populated, the call will return `TRUE` immediately.

HOW TO OVERRIDE

This method can be overridden to modify the contents of a folder. Call the parent method first.

OTHER INFO

Include files: `wpfolder.h`

SEE ALSO

`wpQueryContent` -243, `wpQueryFldrAttr` -245, `wpQueryFldrFlags` -248, `wpUnlockObject` -65

RESTRICTIONS/WARNINGS

- `wpPopulate` will raise the lock count of each object contained in the folder by one (even if the folder is already populated). It is the application's responsibility to use `wpQueryContent` to enumerate the contents and unlock each object when it is no longer needed (see the sample for `wpQueryContent`).

● wpQueryContent WPFolder instance method

Enumerates the contents of a folder.

SYNTAX

WPOBJECT * wpQueryContent(**WPOBJECT *** *Object*, **ULONG** *ulOption*)

PARAMETERS

Object - input

A pointer to the object to use to search for the next object in the content list of the folder. If *ulOption* is not set to `QC_NEXT`, this param-

ter is ignored. If this object is not contained in this folder, wpQueryContent will return FALSE.

ulOption - input

Set to one of the following:

Define	Description
QC_FIRST	0 The first object in the content list will be returned.
QC_NEXT	1 The next object after <i>Object</i> will be returned.
QC_LAST	2 The last object in the content list will be returned.

RETURNS

(nonzero value)—the pointer of an object contained in this folder.

NULL—An error occurred.

HOW TO CALL

Call this method on a populated folder to query its contents. Call wpPopulate first to populate the folder.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpPopulate -242

SAMPLE CODE

The following sample demonstrates how to call wpQueryContent to enumerate the contents of a folder. This can be done from any method override or function where a folder pointer is available.

```
SOMAny *Folder, *Object, *NextObj;
PSZ    pszFileName;
ULONG  ulSize =0;

/* Populate the OS/2 System Folder */
Folder = _wpclsQueryObjectFromPath(_WPFileSystem, "<WP_OS2SYS>");
if (Folder)
{
```

```

_wpQueryRealName(Folder, NULL, &ulSize, TRUE);
if (ulSize)
{
    pszFileName = malloc(ulSize);
    _wpQueryRealName(Folder, pszFileName, &ulSize, TRUE);
    _wpPopulate(Folder, 0, pszFileName, FALSE);
    free(pszFileName);

    /* Loop through the contents of the folder and unlock the
       objects as we go because wpPopulate locked them each once.
    */
    Object = _wpQueryContent(Folder, NULL, QC_FIRST);
    while (Object)
    {
        /* Do something with Object here... */
        /* Get the next object pointer before unlocking Object.
         * There is a small chance that this Object might go to
         * sleep before we can query the next object.
         */
        NextObj = _wpQueryContent(Folder, Object, QC_NEXT);
        _wpUnlockObject(Object);
        Object = NextObj;
    }
}
_wpUnlockObject(Folder);
}

```

● **wpQueryFldrAttr** **WPFolder instance method**

Returns the attributes of a folder's view.

SYNTAX

ULONG wpQueryFldrAttr(**ULONG** *ulView*)

PARAMETERS

ulView - input

The folder view to query. Set *ulView* to one of the following:

Define	Description
OPEN_CONTENTS	1 Icon view
OPEN_TREE	101 Tree view
OPEN_DETAILS	102 Details view

RETURNS

The container attributes for the specified view. These are the view identifier flags used in the `flWindowAttr` field in the `CNRINFO` structure for the container:

Define	Description
CV_TEXT	0x01 Text view. Can be combined with CV_FLOW.
CV_NAME	0x02 Name view. Can be combined with CV_MINI and/or CV_FLOW.
CV_ICON	0x04 Icon view. Can be combined with CV_MINI.
CV_DETAIL	0x08 Details view.
CV_FLOW	0x10 Flowed view.
CV_MINI	0x20 Use mini icons.
CV_TREE	0x40 Tree view. Can be combined with CV_TEXT, CV_NAME, or CV_ICON.

HOW TO CALL

Call this method on any folder to determine the container attributes for its views. Some of these attributes can be set by the user on the View pages of the settings notebook of the folder.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpfolder.h` and `pmstdlg.h`

SEE ALSO

`wpSetFldrAttr` -253

● wpQueryFldrDetailsClass WPFolder instance method

Returns the class used to define the columns of the details view of a folder.

SYNTAX

M_WPObject* wpQueryFldrDetailsClass()

PARAMETERS

none

RETURNS

The pointer to the class object used for the details view of a folder.

HOW TO CALL

Call this method on any folder to determine which class is used to define the columns for its details view. Regardless of which classes of objects are contained in a folder, when a details view is opened, the folder will only display details columns defined by the class object returned in wpQueryFldrDetailsClass. This value can be set by the user on the third View page in the settings notebook of the folder.

HOW TO OVERRIDE

This method is generally not overridden. Call wpSetFldrDetailsClass to change the class object.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpSetFldrDetailsClass -254

RESTRICTIONS/WARNINGS

- Note that the object returned from this method is a *class* object pointer not an *instance* pointer. Do not call instance methods with this pointer.

● wpQueryFldrFlags WPFolder instance method

Returns the flags describing the current state of a folder.

SYNTAX

ULONG wpQueryFldrFlags()

PARAMETERS

none

RETURNS

This object's current folder flags. See wpModifyFldrFlags for the list of possible folder flags.

HOW TO CALL

Call this method on any folder to determine its current folder flags. The FOI_WORKAREA flag is persistent when the folder goes dormant. The populate flags are not.

HOW TO OVERRIDE

This method is generally not overridden. Call wpSetFldrFlags or wpModifyFldrFlags (Warp) to change the flags.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpModifyFldrFlags -241, wpSetFldrFlags -255

RESTRICTIONS/WARNINGS

- A folder will have the FOI_POPULATEDWITHFOLDERS flag but not the FOI_POPULATEDWITHALL flag when wpPopulate is called with the fFoldersOnly flag set to TRUE. The tree view for folders calls wpPopulate in this fashion.

● wpQueryFldrFont WPFolder instance method

Returns the font used for the text in a folder view.

SYNTAX

PSZ wpQueryFldrFont(**ULONG** *ulView*)

PARAMETERS

ulView - input

The view to query. This can be one of the following:

Define		Description
OPEN_CONTENTS	1	Icon view
OPEN_TREE	101	Tree view
OPEN_DETAILS	102	Details view

RETURNS

The pointer to the string with the font name and point size for this view. The format is the point size, followed by a period, followed by the font name. For example, 10 point Helv would be "10.Helv".

HOW TO CALL

Call this method to determine the font used for the text in a folder's view. This value can be set by the user on the View pages in the settings notebook.

HOW TO OVERRIDE

This method is generally not overridden. Call `wpSetFldrFont` to change the font for a view.

OTHER INFO

Include files: `wpfolder.h`

SEE ALSO

`wpSetFldrFont` -256

RESTRICTIONS/WARNINGS

- Note that different folder views can have different fonts assigned to them, but multiple occurrences of the same view will have the same font.

● wpQueryFolder WPObject instance method (Warp only)

Returns the folder that contains an object.

SYNTAX

WPObject* wpQueryFolder()

PARAMETERS

none

RETURNS

The pointer to the folder that contains the object.

HOW TO CALL

Call this method on any object to determine its containing folder.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpobject.h

● wpQueryNextIconPos WPFolder instance method

Returns the next icon position for an object being inserted into the icon view of a folder.

SYNTAX

PPOINTL wpQueryNextIconPos ()

PARAMETERS

none

RETURNS

A pointer to a POINTL structure (pg. 137) that contains the x and y coordinates for the next available slot in a nongridDED icon view.

HOWTO CALL

Call this method on any folder to determine the next position for an object to be placed in the folder's nongridDED icon view.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpfolder.h` and `os2def.h`

● wpSetDetailsColumnVisibility	WPFolder instance method (Warp only)
---------------------------------------	---

Hides or unhides a details data column.

SYNTAX

BOOL wpSetDetailsColumnVisibility (**ULONG** *index*, **BOOL** *Visible*)

PARAMETERS

index - input

The index of the details data column.

Visible - input

If set to TRUE, the column will be visible. If FALSE, the column will be hidden.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOWTO CALL

Call this method at any time to set a column in a folder's details view as visible or invisible. The folder's details view columns are determined by the folder's details class. This class can be queried with `wpQueryFldrDetailsClass`.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpfolder.h`

SEE ALSO

wpclsQueryDetailsInfo - 123, wpIsDetailsColumnVisible - 239,
wpQueryFldrDetailsClass - 247

● wpSetFldrAttr WPFolder instance method

Sets the attributes of a folder's view.

SYNTAX

BOOL wpSetFldrAttr(**ULONG** *Attr*, **ULONG** *ulView*)

PARAMETERS

Attr - input

The container attributes to set for the specified view. These are the flags used in the *flWindowAttr* field in the *CNRINFO* structure for the container. Only specify those flags relevant for the view. For example, do not use the *CV_DETAILS* flag if *ulView* is set to *OPEN_TREE*. See *wpQueryFldrAttr* for a list of attribute flags.

ulView - input

The folder view for which to set the attributes. Set *ulView* to one of the following:

Define	Description	
OPEN_CONTENTS	1	Icon view
OPEN_TREE	101	Tree view
OPEN_DETAILS	102	Details view

RETURNS

TRUE—The attributes were set successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method on any folder to set the container attributes for its views.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfolder.h and pmstdlg.h

SEE ALSO

wpQueryFldrAttr -245

● wpSetFldrDetailsClass WPFolder instance method

Sets the class used to define the columns of the details view of a folder.

SYNTAX

BOOL wpSetFldrDetailsClass(**M_WPObject** * *Class*)

PARAMETERS

Class - input

The class object pointer for the desired class.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method on any folder to specify which class is used to define the columns for its details view. Regardless of which classes of objects are contained in a folder, when a details view is opened, the folder will only display details columns defined by the class object specified by this method. This value can be set by the user on the third View page in the settings notebook of the folder.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpQueryFldrDetailsClass -247

RESTRICTIONS/WARNINGS

- Note that the *Class* parameter is a *class* object pointer not an *instance* pointer.

● **wpSetFldrFlags** **WPFolder instance method**

Sets the flags that describe the current state of a folder.

SYNTAX

BOOL wpSetFldrFlags(**ULONG** *ulFlags*)

PARAMETERS

ulFlags - input

This object's current folder flags. See wpModifyFldrFlags for a list of possible flags.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method on any folder to set its current folder flags. The FOI_WORKAREA flag is persistent when the folder goes dormant. The populate flags are not. Since there are many folder flags that the system uses, always call wpQueryFldrFlags first to preserve those flags. In Warp, wpModifyFldrFlags is the preferred method for changing folder flags.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpModifyFldrFlags -241, wpQueryFldrFlags -248

● wpSetFldrFont WPFolder instance method

Sets the font used for the text in a folder view.

SYNTAX

BOOL wpSetFldrFont(**PSZ** *pszFont*, **ULONG** *ulView*)

PARAMETERS

pszFont - input

The pointer to the string with the font name and point size. The format is the point size, followed by a period, followed by the font name. For example, 10 point Helv would be "10.Helv".

ulView - input

The view for which to set the font. This can be one of the following

Define		Description
OPEN_CONTENTS	1	Icon view
OPEN_TREE	101	Tree view
OPEN_DETAILS	102	Details view

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method to set the font used for the text in a folder's view. This value can be set by the user on the View pages in the settings notebook.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpQueryFldrFont -248

RESTRICTIONS/WARNINGS

- Note that different folder views can have different fonts assigned to them, but multiple occurrences of the same view will have the same font.

●	wpSetFldrSortClass	WPFolder instance method (Warp only)
---	---------------------------	---

Sets the sort class for a folder.

SYNTAX

BOOL wpSetFldrSortClass(**M_WPObject** * *Class*)

PARAMETERS

Class - input

The pointer to the class object to use for the sort functionality of the folder.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method on any folder to specify which class is to be used to define the columns used for sorting. Regardless of which classes of objects are contained in a folder, the sort pulldown in the pop-up will only contain the sortable details columns defined by this class object. This class can be set by the user on the Sort page in the settings notebook of the folder.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpQueryFldrSortClass -250

RESTRICTIONS/WARNINGS

- Note that the *Class* parameter is a *class* object pointer not an *instance* pointer.

● wpSetSortAttribAvailable	WPFolder instance method (Warp only)
-----------------------------------	---

Adds or removes a sort attribute for a folder's Sort pulldown in the pop-up menu.

SYNTAX

BOOL wpSetSortAttribAvailable (**ULONG** *index*, **BOOL** *Available*)

PARAMETERS

index - input

The index of the details data column.

Available - input

If set to TRUE, the details column will be available for sorting. If set to FALSE, the column will not be available.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to specify which sort attributes are available to the user in the pop-up menu of a folder. These sort attributes are those defined by the folder sort class. This class can be queried with wpQueryFldrSortClass.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpclsQueryDetailsInfo -123, wpIsSortAttribAvailable -240,
wpQueryFldrSortClass -250

Class Methods

wpclsQueryActiveDesktop (Warp) returns the object pointer of the active desktop (pg. 259).

wpclsQueryActiveDesktopHWND (Warp) returns the window handle of the active desktop (pg. 260).

wpclsQueryIconDataN (Warp) returns the data of the open icon for a class (pg. 260).

wpclsQueryIconN (Warp) returns the handle of the open icon for a class (pg. 262).

wpclsQueryOpenFolders enumerates all the folders currently open in the system (pg. 262).

wpclsInsertMultipleObjects (Warp) inserts multiple objects into the specified container (pg. 264).

wpclsRemoveObjects (Warp) removes multiple objects from the specified container (pg. 265).

● wpclsQueryActiveDesktop	WPDesktop class method (Warp only)
----------------------------------	---

Returns the object pointer of the active desktop.

SYNTAX

WPObjct * wpclsQueryActiveDesktop()

PARAMETERS

none

RETURNS

The object pointer of the active desktop.

HOW TO CALL

Call this method at any time on the `_WPDesktop` class object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpdesk.h

SEE ALSO:

wpclsQueryActiveDesktopHWND -260

-
- | | |
|---------------------------------------|---|
| ● wpclsQueryActive DesktopHWND | WPDesktop class
method (Warp only) |
|---------------------------------------|---|
-

Returns the window handle of the active desktop.

SYNTAX

HWND wpclsQueryActiveDesktopHWND()

PARAMETERS

none

RETURNS

The frame window handle of the active desktop.

HOW TO CALL

Call this method at any time on the _WPDesktop class object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpdesk.h

SEE ALSO

wpclsQueryActiveDesktop -259

-
- | | |
|------------------------------|--|
| ● wpclsQueryIconDataN | WPFolder class method
(Warp only) |
|------------------------------|--|
-

Returns the data of the open icon of a class.

SYNTAX

ULONG wpclsQueryIconDataN (**PICONINFO** *conInfo*, **ULONG** *ulIconIndex*)

PARAMETERS

pIconInfo - input/output

A pointer to an **ICONINFO** structure (pg. 136) containing the requested icon data. If **NULL** is specified, the required size for the buffer is returned.

ulIconIndex - input

The index for the requested icon. Only the number 1 is currently supported.

RETURNS

(nonzero value)—The size of the icon data for this class.

0—An error occurred.

HOW TO CALL

Call this method at any time to obtain the icon data for the icon displayed when a folder is opened. This method is also called by the system to determine this icon data.

HOW TO OVERRIDE

Override this method to specify the default icon data for your class. If *pIconInfo* is **NULL**, just return the required size for the data. If the **ICON_DATA** format will be used, include the size of the *pIconInfo*->*pIconData* buffer as well. If *pIconInfo* is non-**NULL**, fill in the icon data for the class and return the total size. If the data in *pIconInfo* will be in the **ICON_DATA** format, set *pIconInfo*->*pIconData* to (**PVOID**) (*pIconInfo*+1). You do not need to call the parent method.

OTHER INFO

Include file: `wpfolder.h` and `os2def.h`

SEE ALSO

`wpclsQueryIconData-124`, `wpclsQueryIconN -262`

A pointer to the folder object to use to search for the next open folder. If *ulOption* is not set to QC_NEXT, this parameter is ignored.

ulOption - input

Set to one of the following:

Define	Description
QC_FIRST	0 The first open folder will be returned.
QC_NEXT	1 The next open folder after <i>Folder</i> will be returned.
QC_LAST	2 The last open folder will be returned.

fLock - input

TRUE—Increment the lock count on the folder object returned.

FALSE—Do not increment the lock count.

RETURNS

An object pointer for a folder in the system with an open view.

HOW TO CALL

Call this method at any time on the _WPFolder class object to enumerate the folders currently open in the system. This method only returns folders with an open view of OPEN_CONTENTS, OPEN_TREE, or OPEN_DETAILS. If *fLock* is set to TRUE, call wpUnlockObject on the folder object pointers when they are no longer needed.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfolder.h

SEE ALSO

wpUnlockObject -65

SAMPLE CODE

The following sample is a function called QueryOpenFolders that simply enumerates the open folders and displays their titles.

```
#include <wpfolder.h>
```

```
VOID QueryOpenFolders (VOID)
{
```

```

WPFolder *Folder, *NextFolder;
PSZ    pszTitle;
CHAR    szMessage[256];

/* enumerate the open folders in the system */
Folder = _wpclsQueryOpenFolders(_WPFolder, NULL, QC_FIRST, TRUE);
while (Folder)
{
    pszTitle = _wpQueryTitle(Folder);
    /* warning: this code assumes the folder title will not exceed
     * the szMessage buffer. Applications should check the size of
     * the title before storing it in a buffer.
     */
    sprintf(szMessage, "%s is an open folder.", pszTitle);
    WinMessageBox(HWND_DESKTOP, HWND_DESKTOP, szMessage,
                  "Enumerating Open Folders", 0, MB_OK);

    /* get the next open folder first, then unlock this one. */
    NextFolder = _wpclsQueryOpenFolders(_WPFolder, Folder, QC_NEXT,
                                         TRUE);

    _wpUnlockObject(Folder);
    Folder = NextFolder;
}
}

```

● **wpclsInsertMultipleObjects** **WPObject class method**
(Warp only)

Inserts multiple objects into the specified container.

SYNTAX

BOOL wpclsInsertMultipleObjects(**HWND** *hwndCnr*, **PPOINTL**
pptlIcon, **PVOID** *pObjectArray*,
PVOID *pRecordParent*, **ULONG**
NumRecords)

PARAMETERS

hwndCnr - input

The window handle of the container control in which to insert the objects. The container must be created from a thread running on the Workplace Shell process.

pptlIcon - input

A pointer to a POINTL structure (pg. 137) which contains the icon position at which to insert the first object. This pointer must be set, but the POINTL structure can contain zeros if the view for this container does not support icon positions or the next available position should be used.

pObjectArray - input

An array of pointers to MINIRECORDCORE structures for each object to insert. Call wpQueryCoreRecord on an object to determine its record pointer.

pRecordParent - input

A pointer to the parent record for these objects. If *hwndCnr* is not a tree view or these records do not have a parent, set this to NULL.

NumRecords - input

The number of records in *pObjectArray*.

RETURNS

TRUE—The objects were added successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to add multiple objects to a container. This method has the same effect as calling wpCnrInsertObject multiple times, but is faster.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wobject.h and pmstdlg.h

SEE ALSO

wpclsRemoveObjects -265, wpCnrInsertObject -233,
wpCnrRemoveObject -234, wpQueryCoreRecord -112

●	wpclsRemoveObjects	WPObject class method
		(Warp only)

Removes multiple objects from the specified container.

SYNTAX

BOOL wpclsRemoveObjects(**HWND** *hwndCnr*, **PVOID** *pRecordArray*,
ULONG *NumRecords*, **BOOL** *RemoveAll*)

PARAMETERS

hwndCnr - input

The window handle of the container control from which to remove the objects.

pRecordArray - input

An array of pointers to MINIRECORDCORE structures for each object to remove. Call wpQueryCoreRecord on an object to determine its record pointer.

NumRecords - input

The number of records in *pRecordArray*.

RemoveAll - input

If set to TRUE, all objects are removed from *hwndCnr*. Otherwise, only the objects specified in *pRecordArray* are removed.

RETURNS

TRUE—The objects were removed successfully.

FALSE—An error occurred.

HOWTO CALL

Call this method at any time to remove multiple objects from a container. These objects must have been added to the container using either wpCnrInsertObject or wpclsInsertMultipleObjects. This method has the same effect as calling wpCnrRemoveObject multiple times, but is faster.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include file: wpobject.h and pmstdlg.h

SEE ALSO

wpclsInsertMultipleObjects -264, wpCnrInsertObject -233,
wpCnrRemoveObject -234

Folder Objects: Structures

RECORDINSERT

(24 bytes)

cb (ULONG)	0
The size of this structure, including cb itself.	
pRecordOrder (PRECORDCORE)	4
The record after which to insert this record. Specify CMA_FIRST to place the record at the beginning, CMA_LAST to place the record at the end, or the record pointer of any object already in the container.	
pRecordParent (PRECORDCORE)	8
The parent record under which to place the record. This can be NULL and is ignored for nontree views.	
fInvalidateRecord (ULONG)	12
Specify TRUE and the container will be invalidated when the record is inserted. Specify FALSE and the container will not be invalidated.	
zOrder (ULONG)	16
CMA_TOP places the record on top of other objects. CMA_BOTTOM places it under other objects.	
cRecordsInsert (ULONG)	20
The number of root level records to be inserted. This value must be greater than 0.	
Used by: wpCnrInsertObject -233	

●13

Programs and Associations

Program Classes

There are two types of objects that represent programs in the Workplace Shell: program file objects and program objects. The objects that represent the actual executable files are program file objects (WPProgramFile) which are descendants of WPFileSystem. However, users interact primarily on their desktops with program objects (WPProgram) which are descendants of WPAbstract. Program objects are more useful because they can reside anywhere on the desktop and point to a program file anywhere in the file system. Program files, on the other hand, reside in the folder representing the directory where the executable resides in the file system.

WPProgram and WPProgramFile have most of the same methods. Although Workplace Shell does not use multiple inheritance, both program classes define methods with the same names and share common code.

Associations

Workplace supports file associations as they were defined by applications for OS/2 version 1.3. Programs can list which types of data files are supported through an ASSOCTABLE defined in its resource file. When a program object or a program file object is awakened, the object registers the association information stored in the executable with the system. Then a template for each entry from the ASSOCTABLE is created in the Templates folder and each file of the associated type or extension will have the executable listed in the Open pulldown of its pop-up menu. Each program to which the data file is associated is considered to be a view of the data file object.

PM programs can now set associations between programs and data files using WinSetObjectData and the ASSOCTYPE and ASSOCFILTER setup string keywords (see Appendix A). Workplace applications can also set associations with the wpSetAssociationType and wpSetAssociationFilter methods. These mechanisms eliminate the need for the ASSOCTABLE resource.

An alternative to associations would be to create a WPDataFile subclass that overrides wpclsQueryInstanceType (pg. 226) and/or wpclsQueryInstanceFilter (pg. 227) to designate which types of files are of this class. Instances of such a subclass could have a pop-up menu item that would run the executable with the data file as a parameter. Another alternative is for the user to simply use the Menu page of the settings notebook of file system objects to build his or her own associations.

Restrictions and Warnings

- Menu items in a data file's pop-up menu can be removed through the use of the wpQueryAssociatedProgram method.
- When a program is opened as a result of the user opening an associated file, dragging a file onto the program, or summoning the program from the menu of another object, the program object is not notified that an open is occurring (the wpOpen method is not called).

Instance Methods

wpConfirmKeepAssoc (Warp) confirms whether file extensions of data files can be renamed (pg. 270).

wpQueryAssociatedFileIcon (Warp) returns the icon of the default associated program object of a data file (pg. 271).

wpQueryAssociatedProgram returns the associated program of a data file given the view's pop-up menu ID (pg. 272).

wpQueryAssociationFilter returns the list of file name filters indicating which data files are associated with a program (pg. 275).

wpQueryAssociationType returns the list of file types indicating which data files are associated with a program (pg. 276).

wpQueryProgDetails returns the program details for an object (pg. 277).

wpQueryScreenGroupID (Warp) returns the screen group ID for a running program (pg. 279).

wpSetAssociatedFileIcon sets the icon of a data file to be the icon of its associated program (pg. 280).

wpSetAssociationFilter sets the list of file name filters indicating which data files are associated with a program (pg. 281).

wpSetAssociationType sets the list of file types indicating which data files are associated with a program (pg. 282).

wpSetProgDetails sets the program details for an object (pg. 283).

●	wpConfirmKeepAssoc	WPFileSystem instance method (Warp only)
---	---------------------------	---

Confirms whether file extensions of data files can be renamed.

SYNTAX

ULONG wpConfirmKeepAssoc()

PARAMETERS

none

RETURNS

The response from the user:

Define	Description
KEEP_RENAMEFILESWITHEXT	1 Rename the extension of the file and add the association to the pop-up menu.
DISCARD_RENAMEFILESWITHEXT	2 Rename the extension of the file and discard the association.
CANCEL_RENAMEFILESWITHEXT	3 Do not rename the extension of the file.

HOW TO CALL

This method can be called on any file object to request confirmations from the user when the extension of a file with associations is about to be changed.

HOW TO OVERRIDE

This method can be overridden to change its default behavior.

OTHER INFO

Include files: `wpfsys.h`

● wpQueryAssociatedFileIcon	WPDataFile instance method (Warp only)
------------------------------------	---

Returns the icon of the default associated program object of a data file.

SYNTAX

HPOINTER wpQueryAssociatedFileIcon()

PARAMETERS

none

RETURNS

The handle of the icon representing this file's default associated program.

HOW TO CALL

This method can be called on any data file object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpdataf.h`

SEE ALSO

`wpQueryAssociatedProgram` -272, `wpSetAssociatedFileIcon` -280

● **wpQueryAssociatedProgram** **WPDataFile** instance method

Returns the associated program of a data file given the view's pop-up menu ID.

SYNTAX

WPyObject * wpQueryAssociatedProgram(**ULONG** *ulView*, **PULONG** *pulHowMatched*, **PSZ** *pszMatchString*, **ULONG** *cbMatchString*, **PSZ** *pszDefaultType*)

PARAMETERS

ulView - input

The view of the data file. Note that data file views, other than settings, have the same numeric value as the pop-up menu items in the Open submenu. These view IDs can be queried by sending menu control messages to the *hwndMenu* parameter passed into `wpModifyPopupMenu`. Set *ulView* to `OPEN_RUNNING` to get the default association.

pulHowMatched - output

A pointer to a `ULONG` which, upon return, will contain one of the following values:

Value	Description
0	The view is a program added by the menu page of the settings notebook.
1	The view is an association to a program from a match of a name filter.
2	The view is an association to a program from a match of a type.

pszMatchString - input/output

A pointer to a string which, upon return, will contain the filter or type that caused this file to be associated with the program. The first byte of the string will be set to '\0' if **pulHowMatched* is not 1 or 2.

cbMatchString - input

The size of the *pszMatchString* buffer.

pszDefaultType - input

A pointer to a string containing the default type that should be used to search for matches if the data file has no real type. Set *pszDefaultType* to NULL to prevent the system from using a default type if this object has no type. Set it to -1 to allow the system to use the default type if this object has no type.

RETURNS

(nonzero value)—The pointer to the *WPPProgram* or *WPPProgramFile* object to which this file is associated.

NULL—No association was found or an error occurred.

HOW TO CALL

This method can be called on any data file object to query the program objects associated with the object's views.

HOW TO OVERRIDE

This method can be overridden if a data file subclass defines its own view IDs that represent some type of program association. *wpQueryAssociatedProgram* is called by the system each time a view other than settings is selected from the Open pulldown of the pop-up menu.

OTHER INFO

Include files: `wpdataf.h`

SEE ALSO

wpModifyPopupMenu -199, *wpOpen* -160,
wpQueryAssociationFilter -275, *wpQueryAssociationType* -276,
wpSetAssociationFilter -281, *wpSetAssociationType* -282

RESTRICTIONS/WARNINGS

- This method cannot be called with a NULL pointer for the *pszMatchString* parameter to query the needed size for the buffer. The application must provide a very large buffer (256 should be large enough) to ensure it will fit. If the buffer is not large enough, the

method will still complete successfully, but *pszMatchString* will not be set.

- The return code from this method will be either a *WPProgram* or a *WPProgramFile* instance. Be sure to check the class of the object when calling methods that are specific to one of these classes.
- Be sure to call *wpUnlockObject* on the object pointer returned from this method when it is no longer needed.

SAMPLE CODE

The following sample demonstrates how to call *wpQueryAssociatedProgram* from a *wpOpen* override to determine the associated program that will be opened.

```
SOM_Scope HWND  SOMLINK myf_wpOpen(MyFile *somSelf, HWND hwndCnr,
                                     ULONG ulView, ULONG param)
{
    WPObject *Program;
    ULONG    ulHowMatched;
    CHAR     szMatchString[256], szMessage[256];
    PSZ      pszTitle;

    if (ulView != OPEN_SETTINGS)
    {
        Program = _wpQueryAssociatedProgram(somSelf, ulView,
                                             &ulHowMatched,
                                             szMatchString,
                                             sizeof(szMatchString),
                                             (PSZ) -1);

        if (Program)
        {
            pszTitle = _wpQueryTitle(Program);
            sprintf(szMessage, "%s is associated with this file using"
                    "view %d.",
                    pszTitle, ulView);
            WinMessageBox(HWND_DESKTOP, HWND_DESKTOP, szMessage,
                         "Opening a Data File", 0, MB_OK);
            _wpUnlockObject(Program);
        }
    }
    return (parent_wpOpen(somSelf, hwndCnr, ulView, param));
}
```

●	wpQueryAssociationFilter	WPPProgram(File) instance method
---	---------------------------------	---

Returns the list of file name association filters for a program.

SYNTAX

PSZ wpQueryAssociationFilter ()

PARAMETERS

none

RETURNS

(nonzero value)—The pointer to the string containing a list of association filters for this program. Each filter is separated by a comma. For example, the list of filters for the OS/2 System Editor is "*.TXT;*.DOC".

NULL—This program does not have a list of association filters or an error occurred.

HOWTO CALL

This method can be called at any time on any WPPProgram or WPPProgramFile object. If the source file where this method is called includes both wppgm.h and wppgmf.h and/or the class of the object could be either WPPProgram or WPPProgramFile, call `lookup_wpQueryAssociationFilter` instead of `_wpQueryAssociationFilter`. This will cause SOM to use name lookup to guarantee the correct method is called.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wppgm.h and wppgmf.h

SEE ALSO

`wpQueryAssociatedProgram` -272, `wpQueryAssociationType` -276, `wpSetAssociationFilter` -281, `wpSetAssociationType` -282

● wpQueryAssociationType	WPPProgram(File)
	instance method

Returns the list of file types associated with a program.

SYNTAX

PSZ wpQueryAssociationType ()

PARAMETERS

none

RETURNS

(nonzero value)—The pointer to the string containing a list of associated types for this program. Each type is separated by a comma. For example, the list of types for the OS/2 System Editor is “OS/2 Command File,DOS Command File,Plain Text”.

NULL—This program does not have a list of associated types or an error occurred.

HOWTO CALL

This method can be called at any time on any WPPProgram or WPPProgramFile object. If the source file where this method is called includes both wppgm.h and wppgmf.h and/or the class of the object could be either WPPProgram or WPPProgramFile, call lookup_wpQueryAssociationType instead of _wpQueryAssociationType. This will cause SOM to use name lookup to guarantee the correct method is called.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wppgm.h and wppgmf.h

SEE ALSO

wpQueryAssociatedProgram -272, wpQueryAssociationFilter -275, wpSetAssociationFilter -281, wpSetAssociationType -282

● wpQueryProgDetails WPProgram(File) instance method

Returns the program details for an object.

SYNTAX

BOOL wpQueryProgDetails (**PPROGDETAILS** *pProgDetails*, **PULONG** *pulSize*)

PARAMETERS

pProgDetails - input/output

A pointer to a buffer that will contain the PROGDETAILS (pg. 284) structure and the memory needed for any strings in the program details. Do not just pass the address of a buffer that is only the size of the PROGDETAILS structure. Pass NULL in this parameter and the required size will be returned in *pulSize*.

pulSize - input/output

A pointer to a ULONG which, upon return, will contain the size of the program details for this object. If *pProgDetails* is non-NULL, **pulSize* should be set by the caller to the size of the *pProgDetails* buffer.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time on any WPProgram or WPProgramFile object. If the source file where this method is called includes both `wppgm.h` and `wppgmf.h` and/or the class of the object could be either `WPProgram` or `WPProgramFile`, call `lookup_wpQueryProgDetails` instead of `_wpQueryProgDetails`. This will cause SOM to use name lookup to guarantee the correct method is called. The system will call this method to determine the program details to pass to `WinStartApp` when `wpOpen` is called with *ulView* set to `OPEN_RUNNING`.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wppgm.h`, `wppgmf.h`,
and `pmshl.h`

Define:
`INCL_WINPROGRAMLIST`

SEE ALSO

wpOpen -160, wpSetProgDetails -283

NOTES

Call wpQueryProgDetails before calling wpSetProgDetails to preserve any details you do not wish to modify.

SAMPLE CODE

The following sample demonstrates how to query, modify, and then set the details of a program object. The parameter string for the EPM editor will be changed to point to c:\autoexec.bat. This code can also be used for program file objects.

```
#define INCL_WINPROGRAMLIST

#include <os2.h>

#include <wppgm.h>

VOID ChangeProgDetails(VOID)
{
    PPROGDETAILS pProgDetails;
    ULONG        ulSize;
    WPPProgram    *Program;
    HOBJECT       hProgram;
    CHAR          szParameters[] = "C:\\AUTOEXEC.BAT";

    /* get the object handle for the EPM editor */
    hProgram = WinQueryObject("<WP_EPM>");
    if (hProgram)
    {
        /* get the SOM pointer for the program object */
        Program = _wpclsQueryObject(_WPObject, hProgram);

        /* call wpQueryProgDetails passing NULL for the pointer to get
           the size */
        if (Program && _wpQueryProgDetails(Program, NULL, &ulSize))
        {
            /* allocate the buffer */
            pProgDetails = (PPROGDETAILS) _wpAllocMem(Program, ulSize,
                                                         NULL);

            if (pProgDetails)
            {
```

```

/* do the actual query of the prog details */
if (_wpQueryProgDetails(Program, pProgDetails, &ulSize))
{
    /* pszParameters can be set to a local variable
     * because Workplace will copy the string.
     */
    pProgDetails->pszParameters = szParameters;
    _wpSetProgDetails(Program, pProgDetails);
    _wpSaveDeferred(Program);
}
/* this buffer is no longer needed */
_wpFreeMem(Program, (PBYTE) pProgDetails);
} /* if pProgDetails */
}
if (Program)
    _wpUnlockObject(Program);

} /* if hProgram */
}

```

● **wpQueryScreenGroupID** **WPObject instance method**
(Warp only)

Returns the screen group ID for a running program.

SYNTAX

USHORT wpQueryScreenGroupID(**USHORT** *usPrevSgId*)

PARAMETERS

usPrevSgId - input

If set to 0, the screen group ID of the first running program is returned. If set to an ID previously returned from this method, the ID of the next running program is returned.

RETURNS

The screen group ID for a program running as a view of this object.

HOW TO CALL

This method can be called at any time.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpobject.h`

● wpSetAssociatedFileIcon WPDataFile instance method

Sets the icon of a data file to the icon of its associated program.

SYNTAX

VOID wpSetAssociatedFileIcon ()

PARAMETERS

none

RETURNS

none

HOW TO CALL

Call this method at any time to notify the data file to set its icon to that of its default associated program. The system will call this method on data files when their default associations change. This method will not be called if the data file has a customized icon (see wpSetIconData) or if it is a descendant of WPProgramFile.

HOW TO OVERRIDE

Override this method to prevent the system from setting the icon of a data file to that of its default associated program.

OTHER INFO

Include: `wpdataf.h`

SEE ALSO

wpQueryAssociatedProgram -272, wpSetIconData -120

●	wpSetAssociationFilter	WPPProgram(File) instance method
---	-------------------------------	---

Sets the list of file name association filters of a program.

SYNTAX

BOOL wpSetAssociationFilter (**PSZ** *pszFilter*)

PARAMETERS

pszFilter - input

A pointer to the string containing a list of association filters for this program. Each filter is separated by a comma; for example, "*.C,*.CPP". The system will make a copy of this string; therefore, it can be freed after this call.

RETURNS

TRUE—The association filter was set.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time on any WPPProgram or WPPProgramFile object. If the source file where this method is called includes both wppgm.h and wppgmf.h and/or the class of the object could be either WPPProgram or WPPProgramFile, call `lookup_wpSetAssociationFilter` instead of `_wpSetAssociationFilter`. This will cause SOM to use name lookup to guarantee the correct method is called.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wppgm.h and wppgmf.h

SEE ALSO

wpQueryAssociatedProgram -272, wpQueryAssociationFilter -275,
wpQueryAssociationType -276, wpSetAssociationType -282

●	wpSetAssociationType	WPPProgram(File) instance method
---	-----------------------------	---

Sets the list of file types associated with a program.

SYNTAX

BOOL wpSetAssociationType (**PSZ** *pszType*)

PARAMETERS

pszType - input

The pointer to the string containing a list of associated types for this program. Each type is separated by a comma; for example "C Code, Assembler Code". The system will make a copy of this string; therefore, it can be freed after this call.

RETURNS

TRUE—The type was set.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time on any WPPProgram or WPPProgramFile object. If the source file where this method is called includes both wppgm.h and wppgmf.h and/or the class of the object could be either WPPProgram or WPPProgramFile, call lookup_wpSetAssociationType instead of _wpSetAssociationType. This will cause SOM to use name lookup to guarantee the correct method is called.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wppgm.h and wppgmf.h

SEE ALSO

wpQueryAssociatedProgram -272, wpQueryAssociationFilter -275,
wpQueryAssociationType -276, wpSetAssociationFilter -281

● wpSetProgDetails WPProgram(File) instance method

Sets the program details for an object.

SYNTAX

BOOL wpSetProgDetails (**PPROGDETAILS** *pProgDetails*)

PARAMETERS

pProgDetails - input

A pointer to a buffer containing the PROGDETAILS (pg. 284) structure. The system will make a copy of the structure and all strings it points to; therefore, it can be freed after this call.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time on any WPProgram or WPProgramFile object. If the source file where this method is called includes both wppgm.h and wppgmf.h and/or the class of the object could be either WPProgram or WPProgramFile, call `lookup_wpSetProgDetails` instead of `_wpSetProgDetails`. This will cause SOM to use name lookup to guarantee the correct method is called. The system will pass this data to WinStartApp when `wpOpen` is called with *ulView* set to `OPEN_RUNNING`.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files:	wppgm.h, wppgmf.h, and pmshl.h	Define: INCL_WINPROGRAMLIST
----------------	-----------------------------------	--------------------------------

SEE ALSO

`wpOpen` -160, `wpQueryProgDetails` -277

NOTES

Call `wpQueryProgDetails` before calling `wpSetProgDetails` to preserve any details that you do not wish to modify.

Programs and Associations: Structures

PROGDETAILS

(72 bytes)

Length (ULONG)	0
The size of the PROGDETAILS structure (that is, sizeof(PROGDETAILS)). Note that this size is different than the size parameter for wpQueryProgDetails which includes the size required for the string pointers as well.	
progt (PROGTYPE)	4
A PROGTYPE structure.	
pszTitle (PSZ)	12
The title for the running program. This field can be set to NULL.	
pszExecutable (PSZ)	16
The file name of the program executable. This field can be set to NULL.	
pszParameters (PSZ)	20
The parameter string to pass to the executable. Each parameter is separated by a space. This field can be set to NULL.	
pszStartupDir (PSZ)	24
The working directory for the program. The system will first change directories to the path specified in this field and then execute the program. This field can be set to NULL.	
pszIcon (PSZ)	28
The file name of the icon to set for this program. This field can be set to NULL.	
pszEnvironment (PSZ)	32
A super string of environment strings. Each string is separated by a '\0' character and the entire super string is terminated with two '\0' characters. This field can be set to NULL.	
swpInitial (SWP)	36
The initial size and position of the program.	
Used by: wpQueryProgDetails -277, wpSetProgDetails -283	

PROGTYPE

(8 bytes)

progc (PROGCATEGORY)	0
The program category for this program. This can be one of the following values:	

Define

PROG_DEFAULT	0
PROG_FULLSCREEN	1
PROG_WINDOWABLEVIO	2
PROG_PM	3
PROG_VDM	4
PROG_WINDOWEDVDM	7
PROG_31_STDSEAMLESSVDM	15
PROG_31_STDSEAMLESSCOMMON	16
PROG_31_ENHSEAMLESSVDM	17
PROG_31_ENHSEAMLESSCOMMON	18
PROG_31_ENH	19
PROG_31_STD	20

fbVisible (ULONG)

4

Program visibility flags:

Define

SHE_VISIBLE	0x00
SHE_INVISIBLE	0x01

Used by: PROGDETAILS structure

●14

Disk Objects

The root directories for each drive in the system are instances of the **WPRootFolder** class which is a subclass of **WPFolder**; these objects are never visible to the user. A corresponding **WPDisk** object is placed in the **Drives** folder for each root folder in the system. **WPDisk** is descended from **WPAbstract**, and its instances are used to represent root folder objects and to enable the user to display and manipulate root folders. Since every folder on the desktop must be a subdirectory of the Desktop path (such as `C:\Desktop`), the extra level of indirection provided by **WPDisk** is necessary to make the contents of root folders accessible from the Desktop.

Restrictions and Warnings

- The methods **wpEjectDisk**, **wpLockDrive**, and **wpQueryDriveLockStatus** can only be called on **WPDisk** objects that represent drives that support this functionality.
- Do not call **wpPopulate** on a **WPDisk** object. **WPDisk** is not a descendant of **WPFolder** and does not support this method. Folder manipulation methods should be called on the corresponding **WPRootFolder** object.

Instance Methods

wpEjectDisk ejects removable media from a drive that supports this feature (pg. 287).

wpIsDiskSwapped (Warp) returns whether the removable media of a file system object has been swapped (pg. 288).

wpLockDrive locks or unlocks removable media in a drive that supports this feature (pg. 289).

wpQueryDisk (Warp) returns the disk object representing the drive of a file system object (pg. 290).

wpQueryDriveLockStatus returns the lock status of a drive (pg. 291).

wpQueryLogicalDrive returns the logical drive number this object represents (pg. 292).

wpQueryRootFolder returns the corresponding root folder object for this disk object (pg. 293).

wpSetCorrectDiskIcon (Warp) notifies a disk object to set its icon based on its media type (pg. 293).

● **wpEjectDisk** **WPDisk instance method**

Ejects removable media from a drive that supports this feature.

SYNTAX

ULONG wpEjectDisk()

PARAMETERS

none

RETURNS

0—The call completed successfully.

(nonzero value)—An error occurred. This value is the return code from the call to DosDevIOCtl.

HOWTO CALL

This method can be called at any time on any WPDisk object that supports ejecting removable media. DosDevIOCtl will be called with category IOCTL_DISK and function 40 to perform this action.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpdisk.h

SEE ALSO

wpLockDrive -289, wpQueryDriveLockStatus -291

● **wpIsDiskSwapped** **WPFileSystem instance method**
 (Warp only)

Returns whether the removable media of a file system object has been swapped.

SYNTAX

BOOL wpIsDiskSwapped()

PARAMETERS

none

RETURNS

TRUE—The media has been swapped.

FALSE—No swapping has occurred.

HOW TO CALL

This method can be called at any time on any file system object. Do not call this method on a WPDisk object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpfsys.h`

SEE ALSO

`wpQueryDisk -290`

● **wpLockDrive** **WPDisk instance method**

Locks or unlocks removable media in a drive that supports this feature.

SYNTAX

ULONG wpLockDrive(**BOOL** *fLock*)

PARAMETERS

fLock - input

TRUE—Lock the drive.

FALSE—Unlock the drive.

RETURNS

0—The call completed successfully.

(nonzero value)—An error occurred. This value is the return code from the call to `DosDevIOCtl`.

HOW TO CALL

This method can be called at any time on any WPDisk object that supports locking and unlocking removable media. `DosDevIOCtl` will be called with category `IOCTL_DISK` and function 40 to perform this action.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpdisk.h

SEE ALSO

wpEjectDisk -287, wpQueryDriveLockStatus -291

NOTES

- Call wpQueryDriveLockStatus on a WPDisk object to determine if the drive supports locking and unlocking removable media.

- **wpQueryDisk** **WPFileSystem instance method**
 (Warp only)

Returns the disk object representing the drive of a file system object.

SYNTAX

WPDisk * wpQueryDisk()

PARAMETERS

none

RETURNS

The disk object that represents the drive where the file system object resides.

HOW TO CALL

This method can be called at any time on any file system object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpfsys.h

SEE ALSO

wpIsDiskSwapped -288

RESTRICTIONS/WARNINGS

- Although the prototype for this method in `wpfsys.h` specifies a return type of `WPFileSystem*`, this method returns a `WPDisk*` (both are defined as `SOMAny*`).

● wpQueryDriveLockStatus WPDisk instance method

Returns the lock status of a drive.

SYNTAX

ULONG wpQueryDriveLockStatus(**PULONG** *pullLockStatus*, **PULONG** *pullLockCount*)

PARAMETERS

pullLockStatus - input/output

A pointer to a **ULONG** which, upon return, will contain the following information about the lock status of the drive:

Bits 0 - 1 :	0	The lock and unlock functions are not supported.
	1	The drive is locked.
	2	The drive is unlocked.
	3	The lock function is supported but the status of the drive cannot be determined.
Bit 2 :	0	The drive is empty.
	1	There is media in the drive.
Bits 3-31:	(reserved)	

pullLockCount - currently unsupported

RETURNS

0—The call completed successfully.

(nonzero value)—An error occurred. This value is the return code from the call to `DosDevIOCtrl`.

HOW TO CALL

This method can be called at any time on any `WPDisk` object. `DosDevIOCtrl` will be called with category `IOCTL_DISK` and function 66 to perform this action.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpdisk.h`

SEE ALSO

`wpEjectDisk` -287, `wpLockDrive` -289

● wpQueryLogicalDrive WPDisk instance method

Returns the logical drive number this object represents.

SYNTAX

ULONG wpQueryLogicalDrive()

PARAMETERS

none

RETURNS

The logical drive number of this disk object: 1 = A, 2 = B, and so on.

HOW TO CALL

This method can be called at any time on any WPDisk object to determine which drive it represents.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpdisk.h`

SEE ALSO

`wpQueryRootFolder` -293

● wpQueryRootFolder WPDisk instance method

Returns the corresponding root folder object for a disk object.

SYNTAX

WPRootFolder * wpQueryRootFolder()

PARAMETERS

none

RETURNS

(nonzero value)—The object pointer for the root folder.

HOW TO CALL

This method can be called at any time on any WPDisk object to determine which root folder it represents.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpdisk.h

SEE ALSO

wpQueryLogicalDrive -292

**● wpSetCorrectDiskIcon WPDisk instance method
 (Warp only)**

Notifies a disk object to set its icon based on its media type.

SYNTAX

BOOL wpSetCorrectDiskIcon()

PARAMETERS

none

RETURNS

TRUE—The icon was set.

FALSE—An error occurred.

HOW TO CALL

This method can be called on any disk object.

HOW TO OVERRIDE

Override this method to specify the icon for a disk object. Call `wpQueryLogicalDrive` to determine which drive this disk object represents.

OTHER INFO

Include files: `wpdisk.h`

SEE ALSO

`wpQueryLogicalDrive` -292

●15

Shadow Objects

A shadow is an object whose sole function is to reference other objects. The `WPSHadow` class overrides practically every method defined by `WPObj` and usually calls the same method with the same parameters on the object it is linked to. For example, if you open the settings of a shadow of a folder object, the shadow's `wpOpen` override is called, which then calls `wpOpen` on the folder. Therefore, it is the folder that is actually opening the settings view. The object that a shadow references is called the original object.

Shadows are very useful for users who want to have an object on the desktop that points to an object somewhere else in the file system. It is not possible to edit the path on the file page of a file system object, but it is possible to create a shadow of any object and place it anywhere on the desktop.

Restrictions and Warnings

- Shadows are descendants of WPAbstract and, like other abstract objects, cannot be copied onto the network or removable media.
- Since shadows are WPAbstract descendants, it may not be possible to call methods on a shadow that are supported by the object the shadow is linked to. Only the common WPObject methods or WPSHadow methods can be called on a shadow object.

Instance Methods

wpCreateShadowObjectExt (Warp) creates a shadow of an object using the specified WPSHadow subclass (pg. 296).

wpQueryShadowedObject returns the pointer of the object to which this shadow refers (pg. 298).

wpSetLinkToObject (Warp) sets the link between a shadow and another object (pg. 299).

wpSetShadowTitle sets the title of a shadow to a string different from the title of the original object (pg. 300).

-
- | | |
|----------------------------------|---|
| ● wpCreateShadowObjectExt | WPObject instance
method (Warp only) |
|----------------------------------|---|
-

Creates a shadow of an object using the specified WPSHadow subclass.

SYNTAX

WPObject * wpCreateShadowObjectExt(**WPFolder *** *Folder*, **BOOL** *fLock*,
PSZ *pszSetup*, **M_WPObject ***
shadowClass)

PARAMETERS

Folder - input

The object pointer of the folder in which to place the shadow. Some methods for obtaining folder object pointers are wpclsQueryFolder, wpclsQueryObject, and wpclsQueryObjectFromPath.

fLock - input

TRUE—Increment the lock count of the shadow object returned by this method.

FALSE—Do not increment the lock count.

pszSetup - input

The initial settings for the new shadow object in the form of a setup string. A setup string is a list of keyname value pairs separated by semicolons which is parsed by the object for initialization. Specify NULL to use the object's defaults. See Appendix A for the list of available keyname value pairs for the predefined Workplace Shell classes. For more information on setup strings, see wpSetup (pg. 63).

shadowClass - input

The class of the new shadow object. This must be _WPSshadow or the class object pointer of any subclass of WPSshadow.

RETURNS

(nonzero value)—A pointer to the new shadow object.

NULL—An error occurred.

HOWTO CALL

This method can be called at any time to create a shadow of an object. Use this method instead of wpCreateShadowObject if a setup string is desired or if a different shadow class is required. A USAGE_LINK item will be added to the in-use list of the original object. See Chapter 6 for more information on in-use lists.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpobject.h

SEE ALSO

wpclsQueryFolder -74, wpclsQueryObject -75,
wpclsQueryObjectFromPath -76, wpAddToObjUseList -145,
wpCreateShadowObject -50, wpQueryShadowedObject -298,
wpSetup -63, wpUnlockObject -65

NOTES

Pass TRUE for the *fLock* parameter if you are going to access the shadow object pointer after calling `wpCreateShadowObjectExt`. When the pointer is no longer needed, call `wpUnlockObject`.

● wpQueryShadowedObject WPSshadow instance method

Returns the pointer of the object to which this shadow refers.

SYNTAX

WPObj * wpQueryShadowedObject(**BOOL** *fLock*)

PARAMETERS

fLock - input

TRUE—Increment the lock count of the object returned by this method.

FALSE—Do not increment the lock count.

RETURNS

(nonzero value)—A pointer to the object this shadow points to.

NULL—An error occurred or this shadow is currently not pointing to an object.

HOWTO CALL

This method can be called at any time on any WPSshadow descendant to query its original object.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpshadow.h`

NOTES

Pass TRUE for the *fLock* parameter if you are going to access the object pointer after calling `wpQueryShadowedObject`. When the pointer is no longer needed, call `wpUnlockObject`.

● **wpSetLinkToObject** **WPSHadow instance method**
(Warp only)

Sets the link between a shadow and another object.

SYNTAX

BOOL wpSetLinkToObject(**WPObject** * *FromObject*)

PARAMETERS

FromObject - input

The pointer of the object to which this shadow is to be linked.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpshadow.h

SEE ALSO

wpCreateShadowObject -50, wpCreateShadowObjectExt -296

NOTES

This method does not need to be called on a shadow created with wpCreateShadowObject or wpCreateShadowObjectExt.

● wpSetShadowTitle WPSHADOW instance method

Sets the title of a shadow to be different from the title of the original object

SYNTAX

BOOL wpSetShadowTitle(**PSZ** *pszNewTitle*)

PARAMETERS

pszNewTitle - input

A pointer to a string containing the new title for the shadow object. The system will make a copy of this string; therefore, it can be freed after this call.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time on any WPSHADOW descendant to set its title without modifying the title of the object to which it refers.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpshadow.h

SEE ALSO

wpQueryTitle -118, wpSetTitle -122

RESTRICTIONS/WARNINGS

- If wpSetTitle is called on a shadow object, it changes the title of the shadow as well as the original object.
- There is no method to query the title of a shadow specifically. wpQueryTitle will return the title of the original object.
- If wpSetShadowTitle is called on a shadow object and wpSaveDeferred or wpSaveImmediate is also called, the title will revert back to that of the original object.

●16

Launch Pad Objects

The Launch Pad is a new feature in Warp that enables the user to launch programs quickly and easily. When an object is added to the Launch Pad, a shadow of the object is created in the Nowhere folder and a button is displayed to represent that object. The default view of the object is opened when the user presses the button. Each item in the Launch Pad also has a drawer where more objects can be placed.

The WPLaunchPad class can be instantiated more than once and it can also be subclassed; therefore, many launch pads can exist in the system at once (the system will always open the Launch Pad with the object id of <WP_LAUNCHPAD>). It is also possible to enhance the main Launch Pad in the system by replacing the WPLaunchPad class using WinReplaceObjectClass (pg. 19). To call methods on the default Launch Pad, call WinQueryObject on <WP_LAUNCHPAD> and then wpclsQueryObject to obtain the object pointer.

Restrictions and Warnings

- The methods described in this chapter can only be used by applications developed for OS/2 Warp (version 3.0) or later versions.

Instance Methods

wpQueryActionButtons returns the list of action buttons displayed on the pad (pg. 303).

wpQueryActionButtonStyle returns the style of the action buttons (pg. 304).

wpQueryCloseDrawer returns whether to close a drawer when a contained object is launched (pg. 305).

wpQueryDisplaySmallIcons returns whether the buttons should display small icons (pg. 305).

wpQueryDisplayText returns whether to display text on the buttons in the main pad (pg. 306).

wpQueryDisplayTextInDrawers returns whether to display text on the buttons in the drawers (pg. 307).

wpQueryDisplayVertical returns whether to display the pad vertically (pg. 308).

wpQueryDrawerHWND returns the window handle of a drawer (pg. 308).

wpQueryFloatOnTop returns whether the pad should always be surfaced to the top of the window z-order (pg. 309).

wpQueryHideLaunchPadFrameCtrls returns whether to hide the frame controls on the main pad (pg. 310).

wpQueryObjectList returns the list of objects displayed in the main pad or a specific drawer (pg. 311).

wpRefreshDrawer refreshes a drawer window or the main pad (pg. 312).

wpSetActionButtonStyle sets the style of the action buttons (pg. 313).

wpSetCloseDrawer sets whether to close a drawer when a contained object is launched (pg. 314).

wpSetDisplaySmallIcons sets whether the buttons should display small icons (pg. 314).

wpSetDisplayText sets whether to display text under the buttons in the main pad (pg. 315).

wpSetDisplayTextInDrawers sets whether to display text under the buttons in the drawers (pg. 316).

wpSetDisplayVertical sets whether to display the pad vertically (pg. 317).

wpSetDrawerHWND sets the window handle of a drawer (pg. 317).

wpSetFloatOnTop sets whether the pad should always be surfaced to the top of the window z-order (pg. 318).

wpSetHideLaunchPadFrameCtrls sets whether to hide the frame controls on the main pad (pg. 319).

wpSetObjectListFromHObjects inserts a list of objects into the pad using object handles (pg. 320).

wpSetObjectListFromObjects inserts a list of objects into the pad using object pointers (pg. 321).

wpSetObjectListFromStrings inserts a list of objects into the pad using object IDs or file names (pg. 322).

● **wpQueryActionButtons** **WPLaunchPad instance method**

Returns the list of action buttons displayed on the pad.

SYNTAX

PACTIONS wpQueryActionButtons(**P**ULONG *pulNumActions*)

PARAMETERS

pulNumActions - output

A pointer to the number of actions returned on this call.

RETURNS

A pointer to an array of **ACTIONS** (pg. 323) structures. *pulNumActions* contains the number of **ACTIONS** in the array.

HOW TO CALL

Call this method at any time to receive information describing the action buttons on the main pad.

HOW TO OVERRIDE

This method can be overridden to modify the returned actions. The parent method does not have to be called.

OTHER INFO

Include files: wplnchpd.h

SEE ALSO

wpQueryActionButtonStyle -304, wpSetActionButtonStyle -313

●

wpQueryActionButtonStyle

WPLaunch Pad
instance method

Returns the style of the action buttons.

SYNTAX

ULONG wpQueryActionButtonStyle()

PARAMETERS

none

RETURNS

The style for the action buttons on the main pad. One of the following values can be returned to specify how to display the buttons:

Define	Description
ACTION_BUTTONS_TEXT	0 Display the action buttons as text.
ACTION_BUTTONS_OFF	1 Do not display the action buttons.
ACTION_BUTTONS_MINI	2 Display the action buttons with mini icons.
ACTION_BUTTONS_NORMAL	3 Display the action buttons with normal-sized icons.

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification; however, wpSetActionButtonStyle should be used to change the style of the action buttons.

OTHER INFO

Include files: `wplnchpd.h`

SEE ALSO

`wpQueryActionButtons` -303, `wpSetActionButtonStyle` -313

● wpQueryCloseDrawer WPLaunchPad instance method

Returns whether to close a drawer when a contained object is launched.

SYNTAX

BOOL `wpQueryCloseDrawer()`

PARAMETERS

none

RETURNS

TRUE—Close drawers when their contents are launched.

FALSE—Leave drawers opened.

HOW TO CALL

Call this method at any time. This behavior applies only to the drawers in a launch pad and not the main pad.

HOW TO OVERRIDE

This method can be overridden for notification; however, `wpSetCloseDrawer` should be used to change the behavior.

OTHER INFO

Include files: `wplnchpd.h`

SEE ALSO

`wpSetCloseDrawer` -314

**● wpQueryDisplaySmallIcons WPLaunch Pad
instance method**

Returns whether the buttons should display small icons.

SYNTAX

BOOL wpQueryDisplaySmallIcons()

PARAMETERS

none

RETURNS

TRUE—Display mini icons on the buttons.

FALSE—Display normal-sized icons.

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification; however, wpSetDisplaySmallIcons should be used to change this setting.

OTHER INFO

Include files: wplnchpd.h

SEE ALSO

wpSetDisplaySmallIcons -314

● wpQueryDisplayText WPLaunchPad instance method

Returns whether to display text on the buttons on the main pad.

SYNTAX

BOOL wpQueryDisplayText()

PARAMETERS

none

RETURNS

TRUE—Display the object title text on the buttons on the main pad.

FALSE—Do not display the text.

HOW TO CALL

Call this method at any time. This setting applies only to the buttons on the main pad. Call `wpQueryDisplayTextInDrawers` to query the setting for the drawers.

HOW TO OVERRIDE

This method can be overridden for notification; however, `wpSetDisplayText` should be used to change this setting.

OTHER INFO

Include files: `wplnchpd.h`

SEE ALSO

`wpQueryDisplayTextInDrawers` -307, `wpSetDisplayText` -315, `wpSetDisplayTextInDrawers` -316

●	<code>wpQueryDisplayTextInDrawers</code>	WPLaunch Pad instance method
---	---	---

Returns whether to display text on the buttons in the drawers.

SYNTAX

BOOL `wpQueryDisplayTextInDrawers()`

PARAMETERS

none

RETURNS

TRUE—Display the object title text on the buttons in the drawers.

FALSE—Do not display the text.

HOW TO CALL

Call this method at any time. This setting applies only to the buttons contained in drawers. Call `wpQueryDisplayText` to query the setting for the main pad.

HOW TO OVERRIDE

This method can be overridden for notification; however, `wpSetDisplayTextInDrawers` should be used to change this setting.

OTHER INFO

Include files: `wplnchpd.h`

SEE ALSO

`wpQueryDisplayText` -306, `wpSetDisplayText` -315,
`wpSetDisplayTextInDrawers` -316

● wpQueryDisplayVertical WPLaunchPad instance method

Returns whether to display the pad vertically.

SYNTAX

BOOL `wpQueryDisplayVertical()`

PARAMETERS

none

RETURNS

TRUE—Display the pad vertically and the drawers horizontally.

FALSE—Display the pad horizontally and the drawers vertically. This is the default.

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification; however, `wpSetDisplayVertical` should be used to change the behavior.

OTHER INFO

Include files: `wplnchpd.h`

SEE ALSO

`wpSetDisplayVertical` -317

● wpQueryDrawerHWND WPLaunchPad instance method

Returns the window handle of a drawer.

SYNTAX

HWND wpQueryDrawerHWND(**ULONG** *ulDrawer*)

PARAMETERS

ulDrawer - input

The number representing the drawer to query: 0 indicates the main pad; any other value indicates the index of the specific drawer; 1 is the first, 2 is the second, and so on.

RETURNS

(nonzero value)—The window handle of the main pad (*ulDrawer* is 0) or the window handle for a drawer (*ulDrawer* is greater than 0).

NULLHANDLE—The view is not open (*ulDrawer* is 0) or the drawer is not open (*ulDrawer* is greater than 0).

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification; however, wpSetDrawerHWND should be overridden for notification when a drawer is opened.

OTHER INFO

Include files: wplnchpd.h

SEE ALSO

wpSetDrawerHWND -317

● wpQueryFloatOnTop WPLaunchPad instance method

Returns whether the pad should always be surfaced to the top of the window z-order.

SYNTAX

BOOL wpQueryFloatOnTop()

PARAMETERS

none

RETURNS

TRUE—Always surface the Launch Pad.

FALSE—Allow the pad to be covered by other windows.

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification; however, `wpSetFloatOnTop` should be used to change the behavior.

OTHER INFO

Include files: `wplnchpd.h`

SEE ALSO

`wpSetFloatOnTop` -318

●	<code>wpQueryHideLaunchPadFrameCtrls</code>	<code>WPLaunchPad</code> instance method
---	--	---

Returns whether to hide the frame controls on the main pad.

SYNTAX

BOOL `wpQueryHideLaunchPadFrameCtrls()`

PARAMETERS

none

RETURNS

TRUE—Hide the system menu and title bar on the main pad.

FALSE—Do not hide the frame controls.

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification; however, `wpSetHideLaunchPadFrameCtrls` should be used to change this setting.

OTHER INFO

Include files: wplnchpd.h

SEE ALSO

wpSetHideLaunchPadFrameCtrls -319

● wpQueryObjectList WPLaunchPad instance method

Returns the list of objects displayed in the main pad or a specific drawer.

SYNTAX

HOBJECT * wpQueryObjectList(**ULONG** *ulDrawer*, **PULONG** *pulNumObjects*)

PARAMETERS

ulDrawer - input

The number representing the drawer to query: 0 indicates the main pad; any other value indicates the index of the specific drawer; 1 is the first, 2 is the second, and so on.

pulNumObjects - output

A pointer to the number of object handles returned from this call.

RETURNS

A pointer to an array of object handles representing the objects contained in the main pad (*ulDrawer* is 0) or the drawer (*ulDrawer* is greater than 0).

HOW TO CALL

Call this method at any time. Call wpclsQueryObject to obtain an object pointer from an object handle.

HOW TO OVERRIDE

This method can be overridden; however, wpSetObjectListFromHObjects, wpSetObjectListFromObjects, or wpSetObjectListFromStrings should be used to change the contents of a launch pad.

OTHER INFO

Include files: wplnchpd.h

SEE ALSO

wpclsQueryObject -75, wpSetObjectListFromHObjects -320,
wpSetObjectListFromObjects -321, wpSetObjectListFromStrings -322

NOTES

The objects returned from this method are the shadow objects created by the Launch Pad, not the objects specified by the wpSetObjectListFromXxx call.

● wpRefreshDrawer WPLaunchPad instance method

Refreshes a drawer window or the main pad.

SYNTAX

VOID wpRefreshDrawer(**ULONG** *ulDrawer*)

PARAMETERS

ulDrawer - input

The number representing the drawer to refresh: 0 indicates the main pad; any other value indicates the index of the specific drawer; 1 is the first, 2 is the second, and so on.

RETURNS

none

HOWTO CALL

Call this method at any time to force the main pad or a drawer to be updated after changes have been made to its contents. While processing this method, the system will call wpSetDrawerHWND to set the handle to NULLHANDLE while refreshing, and then call wpSetDrawerHWND to reset the handle.

HOWTO OVERRIDE

This method can be overridden for notification when the main pad or a drawer is being refreshed.

OTHER INFO

Include files: wplnchpd.h

SEE ALSO

wpSetDrawerHWND -317

● wpSetActionButtonStyle	WPLaunchPad instance method
---------------------------------	--

Sets the style of the action buttons.

SYNTAX

BOOL wpSetActionButtonStyle(**ULONG** *ulStyle*)

PARAMETERS

ulStyle - input

The style for the action buttons on the main pad. One of the following values can be used to specify how to display the buttons:

Define	Description
ACTION_BUTTONS_TEXT	0 Display the action buttons as text.
ACTION_BUTTONS_OFF	1 Do not display the action buttons.
ACTION_BUTTONS_MINI	2 Display the action buttons with mini icons.
ACTION_BUTTONS_NORMAL	3 Display the action buttons with normal-sized icons.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification when the action button style has been changed.

OTHER INFO

Include files: wplnchpd.h

SEE ALSO

wpQueryActionButtons -303, wpQueryActionButtonStyle -304

● wpSetCloseDrawer WPLaunchPad instance method

Sets whether to close a drawer when a contained object is launched.

SYNTAX

VOID wpSetCloseDrawer(**BOOL** *fState*)

PARAMETERS

fState - input

TRUE—Close drawers when their contents are launched.

FALSE—Leave drawers opened.

RETURNS

none

HOW TO CALL

Call this method at any time. This behavior applies only to the drawers in a launch pad and not the main pad.

HOW TO OVERRIDE

This method can be overridden for notification when this behavior is changed.

OTHER INFO

Include files: wplnchpd.h

SEE ALSO

wpQueryCloseDrawer -305

● wpSetDisplaySmallIcons WPLaunchPad instance method

Sets whether the buttons should display small icons.

SYNTAX

VOID wpSetDisplaySmallIcons(**BOOL** *fState*)

PARAMETERS

fState - input

TRUE—Display mini icons on the buttons.

FALSE—Display normal-sized icons.

RETURNS

none

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification when this setting is changed.

OTHER INFO

Include files: `wplnchpd.h`

SEE ALSO

`wpQueryDisplaySmallIcons` -305

● **wpSetDisplayText** **WPLaunchPad instance method**

Sets whether to display text on the buttons on the main pad.

SYNTAX

VOID `wpSetDisplayText(BOOL fState)`

PARAMETERS

fState - input

TRUE—Display the object title text on the buttons on the main pad.

FALSE—Do not display the text.

RETURNS

none

HOW TO CALL

Call this method at any time. This setting applies only to the buttons on the main pad. Call `wpSetDisplayTextInDrawers` to change the setting for the drawers.

HOW TO OVERRIDE

This method can be overridden for notification when this setting is changed.

OTHER INFO

Include files: wplnchpd.h

SEE ALSO

wpQueryDisplayText -306, wpQueryDisplayTextInDrawers -307,
wpSetDisplayTextInDrawers -316

● wpSetDisplay TextInDrawers	WPLaunchPad instance method
-------------------------------------	--

Sets whether to display text on the buttons in the drawers.

SYNTAX

VOID wpSetDisplayTextInDrawers(**BOOL** *fState*)

PARAMETERS

fState - input

TRUE—Display the object title text on the buttons contained in the drawers.

FALSE—Do not display the text.

RETURNS

none

HOW TO CALL

Call this method at any time. This setting applies only to the buttons contained in drawers. Call wpSetDisplayText to set the setting for the main pad.

HOW TO OVERRIDE

This method can be overridden for notification when this setting is changed.

OTHER INFO

Include files: wplnchpd.h

SEE ALSO

wpQueryDisplayText -306, wpQueryDisplayTextInDrawers -307,
wpSetDisplayText -315

● wpSetDisplayVertical WPLaunchPad instance method

Sets whether to display the pad vertically.

SYNTAX

VOID wpSetDisplayVertical(**BOOL** *fState*)

PARAMETERS

fState - input

TRUE—Display the pad vertically and the drawers horizontally.

FALSE—Display the pad horizontally and the drawers vertically. This is the default.

RETURNS

none

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification when this behavior is changed.

OTHER INFO

Include files: wplnchpd.h

SEE ALSO

wpQueryDisplayVertical -308

● wpSetDrawerHWND WPLaunchPad instance method

Sets the window handle of a drawer.

SYNTAX

VOID wpSetDrawerHWND(**ULONG** *ulDrawer*, **HWND** *hwnd*)

PARAMETERS

ulDrawer - input

The number representing the drawer to query: 0 indicates the main pad; any other value indicates the index of the specific drawer; 1 is the first, 2 is the second, and so on.

hwnd - input

The window handle of the main pad (*ulDrawer* is 0) or the window handle for the drawer (*ulDrawer* is greater than 0).

RETURNS

none

HOW TO CALL

This method is generally called by the system.

HOW TO OVERRIDE

This method can be overridden for notification when a drawer is opened.

OTHER INFO

Include files: wplnchpd.h

SEE ALSO

wpQueryDrawerHWND -308

● **wpSetFloatOnTop** **WPLaunchPad instance method**

Sets whether the pad should always be surfaced to the top of the window z-order.

SYNTAX

VOID wpSetFloatOnTop(**BOOL** *fState*)

PARAMETERS

fState - input

TRUE— Always surface the Launch Pad.

FALSE—Allow the pad to be covered by other windows.

RETURNS

none

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification when the behavior is changed.

OTHER INFO

Include files: `wplnchpd.h`

SEE ALSO

`wpQueryFloatOnTop` -309

NOTES

If more than one launch pad is set with the float-on-top behavior, the result will be unpredictable.

● wpSetHideLaunch PadFrameCtrls	WPLaunchPad instance method
--	--

Sets whether to hide the frame controls on the main pad.

SYNTAX

VOID `wpSetHideLaunchPadFrameCtrls(BOOL fState)`

PARAMETERS

fState - input

TRUE—Hide the system menu and title bar on the main pad.

FALSE—Do not hide the frame controls.

RETURNS

none

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification when this setting is changed.

OTHER INFO

Include files: `wplnchpd.h`

SEE ALSO

`wpQueryHideLaunchPadFrameCtrls -310`

● wpSetObjectList FromHObjects WPLaunchPad instance method

Inserts a list of objects into the pad using object handles.

SYNTAX

```
BOOL wpSetObjectListFromHObjects(ULONG ulDrawer, ULONG
                                ulNumObjects, HOBJECT *phob-
                                jects , ULONG ulAfter)
```

PARAMETERS

ulDrawer - input

The number representing the drawer to set: 0 indicates the main pad; any other value indicates the index of the specific drawer; 1 is the first, 2 is the second, and so on.

ulNumObjects - output

A pointer to the number of object handles in *phobjects*.

phobjects - input

A pointer to an array of object handles that represents the objects to be inserted into the pad.

ulAfter - input

The place after which to place the objects. Set this to `ADD_OBJECTS_FIRST` to place them at the front, `ADD_OBJECTS_LAST` to place them at the end, or a number greater than 0 specifying the index of the object after which to place the objects.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification when objects are added to the Launch Pad.

OTHER INFO

Include files: `wplnchpd.h`

SEE ALSO

`wpQueryObjectList` -311, `wpSetObjectListFromObjects` -321,
`wpSetObjectListFromStrings` -322

● **wpSetObjectList FromObjects** **WPLaunchPad** instance method

Inserts a list of objects into the pad using object pointers.

SYNTAX

```
BOOL wpSetObjectListFromObjects(ULONG ulDrawer, ULONG
                                ulNumObjects, WPObject **objects,
                                ULONG ulAfter)
```

PARAMETERS

ulDrawer - input

The number representing the drawer to set: 0 indicates the main pad; any other value indicates the index of the specific drawer; 1 is the first, 2 is the second, and so on.

ulNumObjects - output

A pointer to the number of object pointers in *objects*.

objects - input

A pointer to an array of object pointers to be inserted into the pad.

ulAfter - input

The place after which to place the objects. Set this to `ADD_OBJECTS_FIRST` to place them at the front, `ADD_OBJECTS_LAST` to place them at the end, or a number greater than 0 specifying the index of the object after which to place the objects.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification when objects are added to the Launch Pad.

OTHER INFO

Include files: `wplnchpd.h`

SEE ALSO

`wpQueryObjectList` -311, `wpSetObjectListFromHObjects` -320,
`wpSetObjectListFromStrings` -322

●	wpSetObjectListFromStrings	WPLaunchPad instance method
---	-----------------------------------	--

Inserts a list of objects into the pad using object IDs or file names.

SYNTAX

BOOL wpSetObjectListFromStrings(**ULONG** *ulDrawer*, **PSZ** *pszSetup*,
ULONG *ulAfter*)

PARAMETERS

ulDrawer - input

The number representing the drawer to set: 0 indicates the main pad; any other value indicates the index of the specific drawer; 1 is the first, 2 is the second, and so on.

pszSetup - input

A super string containing object IDs and/or fully qualified paths of objects to insert. Each item is separated by a '\0' character and the entire string is terminated with two '\0' characters.

ulAfter - input

The place after which to place the objects. Set this to `ADD_OBJECTS_FIRST` to place them at the front, `ADD_OBJECTS_LAST` to

place them at the end, or a number greater than 0 specifying the index of the object after which to place the objects.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method can be overridden for notification when objects are added to the Launch Pad.

OTHER INFO

Include files: `wplnchpd.h`

SEE ALSO

`wpQueryObjectList` -311, `wpSetObjectListFromHObjects` -320,
`wpSetObjectListFromObjects` -321

Launch Pad Objects: Structures

ACTIONS

(8 bytes)

pszTitle (PSZ)

0

The title for the action button.

ulMenuId (ULONG)

4

The menu ID for this action. When an action button is pressed, the menu ID is sent to the desktop via `wpMenuItemSelected`. This should be a valid desktop menu ID (defined in `wpobject.h`). If this is set to something other than a valid desktop menu ID, the `WPDesktop` class must be replaced to handle the new value in the `wpMenuItemSelected` override. Choose from the following list of desktop menu IDs:

Define	Menu item
WPMENUID_FIND	8 Find
WPMENUID_TREE	123 Tree view
WPMENUID_ICON	303 Icon view
WPMENUID_DETAILS	304 Details view
WPMENUID_SHUTDOWN	704 Shutdown
WPMENUID_LOCKUP	705 Lockup
WPMENUID_SYSTEMSETUP	713 System setup

Used by: wpQueryActionButtons -303

●17

Palette Objects

A palette is an object that presents a selection of cells that can be dropped on a window to perform an action. For example, the palette view of the Color Palette shows a grid of color circles that can be edited or dropped on a window to change its color.

A subclass of `WPPalette` can be written to create user-defined palettes. The `WPPalette` class provides a palette window with empty cells, while the subclass paints the cells and provides the edit dialog and drag behavior. Each cell can be assigned user-defined data; `WPPalette` saves this cell data in `wpSaveState`.

Restrictions and Warnings

- There is no method to set the help panel to display when help is requested from a palette window. `wpQueryPaletteHelp` returns only a help panel ID which will be used with the default Workplace help library; therefore other libraries cannot be specified. To provide help, override `wpOpen` and subclass the client window created by the parent class (the client can be obtained by calling `WinWindowFromID` on the frame using the `FID_CLIENT` ID). From the subclass window procedure, process the `WM_HELP` message.
- There is no method for changing the instruction text of a palette window.

Instance Methods

wpDragCell notifies a palette that the user has started to drag a cell (pg. 327).

wpEditCell notifies a palette that the user has requested to edit a cell (pg. 328).

wpPaintCell paints a cell in a palette (pg. 329).

wpRedrawCell notifies a palette that a cell needs to be redrawn (pg. 330).

wpQueryPaletteInfo returns information about a palette object (pg. 330).

wpSelectCell (Warp) selects a cell in a palette view (pg. 331).

wpSetPaletteInfo sets information about a palette object (pg. 332).

wpSetupCell sets the class specific data on a cell of a palette object (pg. 333).

● **wpDragCell** **WPPalette instance method**

Notifies a palette that the user has started to drag a cell.

SYNTAX

BOOL wpDragCell(**PCELL** *pCell*, **HWND** *hwndPal*, **POINTL** *ptlDrag*)

PARAMETERS

pCell - input

A pointer to the CELL (pg. 335) structure of the cell under the cursor when the user began the drag.

hwndPal - input

The window handle for the open palette window.

ptlDrag - input

A pointer to a POINTL (pg. 137) structure which contains the coordinates of the mouse pointer when the drag began. These coordinates are in relation to the palette window.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method is generally only called by the system when the palette window receives the WM_BEGINDRAG message.

HOW TO OVERRIDE

Override this method to provide the drag functionality for a WPPalette subclass. WPPalette subclasses do not use the PM Drag facility but instead capture the mouse pointer. In this override, capture the mouse and use an invisible window to process the messages. When the window receives the WM_BUTTON2UP message, perform the action defined by this palette subclass on the window under the mouse pointer and stop capturing the mouse. The parent method takes no action and returns FALSE.

OTHER INFO

Include files: wppalet.h and os2def.h

SEE ALSO

wpEditCell -328, wpPaintCell -329, wpRedrawCell -330,
wpSetupCell -333

● wpEditCell WPPalette instance method

Notifies a palette that the user has requested to edit a cell.

SYNTAX

BOOL wpEditCell(**PCELL** *pCell*, **HWND** *hwndPal*)

PARAMETERS

pCell - input

A pointer to the CELL (pg. 335) structure of the cell to edit.

hwndPal - input

The window handle for the open palette window.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method is generally only called by the system when the user double-clicks on a cell or selects the edit button.

HOW TO OVERRIDE

Override this method to provide the interface for the user to edit the cell. The parent method takes no action and returns FALSE.

OTHER INFO

Include files: wppalet.h and os2def.h

SEE ALSO

wpDragCell -327, wpPaintCell -329, wpRedrawCell -330,
wpSetupCell -333

● wpPaintCell WPPalette instance method

Paints a cell in a palette.

SYNTAX

BOOL wpPaintCell(**PCELL** *pCell*, **HPS** *hps* , **PRECTL** *prcl*, **BOOL** *fHilite*)

PARAMETERS

pCell - input

A pointer to the CELL (pg. 335) structure of the cell that must be painted.

hps - input

The presentation space handle for the palette window.

prcl - input

A pointer to a RECTL structure (pg. 336) that contains the coordinates for the cell.

fHilite - input

TRUE—The cell is currently selected.

FALSE—This cell is not currently selected.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method is generally only called by the system when a cell must be painted in response to a WM_PAINT message or the wpRedrawCell method.

HOW TO OVERRIDE

Override this method to paint the cells of the palette. The parent method will fill the cell with SYSCLR_WINDOW if *fHilite* is FALSE or SYSCLR_HIGHLIGHTBACKGROUND if *fHilite* is TRUE. If the parent method is called, it should be called first.

OTHER INFO

Include files: wppalet.h and os2def.h

SEE ALSO

wpDragCell -327, wpEditCell -328, wpRedrawCell -330,
wpSetupCell -333

● wpRedrawCell WPPalette instance method

Notifies a palette that a cell needs to be redrawn.

SYNTAX

BOOL wpRedrawCell(**PCELL** *pCell*)

PARAMETERS

pCell - input

A pointer to the CELL (pg. 335) structure of the cell to redraw.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOWTO CALL

This method can be called at any time to invalidate the region of a cell to force it to redraw. This will cause the wpPaintCell method to be called.

HOWTO OVERRIDE

This method is generally not overridden. Override wpPaintCell to actually paint the cell.

OTHER INFO

Include files: wppalet.h

SEE ALSO

wpDragCell -327, wpEditCell -328, wpPaintCell -329, wpSetupCell -333

● wpQueryPaletteInfo WPPalette instance method

Returns information about a palette object.

SYNTAX

BOOL wpQueryPaletteInfo(**PPALINFO** *pPalInfo*)

PARAMETERS

pPalInfo - input/output

A pointer to a PALINFO structure (pg. 335) which, upon return, will be filled with information about this palette object.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method any time to retrieve information about a palette object. This is useful in determining the number of cells to set up with wpSetupCell.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wppalet.h

SEE ALSO

wpSetPaletteInfo -332, wpSetupCell -333

● wpSelectCell WPPalette instance method (Warp only)

Selects a cell in a palette view.

SYNTAX

VOID wpSelectCell(**HWND** *hwndPal*, **PCELL** *pCell*)

PARAMETERS

hwndPal - input

The window handle of the Palette view.

pCell - input

A pointer to the CELL (pg. 335) structure of the cell to select.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time but is generally called by the system when the user selects a cell.

HOW TO OVERRIDE

Override this method for notification when a cell has been selected. `wpPaintCell` will also be called to redraw the cell.

OTHER INFO

Include files: `wppalet.h`

SEE ALSO

`wpPaintCell` -329, `wpSetupCell` -333

● wpSetPaletteInfo WPPalette instance method

Sets information about a palette object.

SYNTAX

BOOL `wpSetPaletteInfo`(**PPALINFO** *pPalInfo*)

PARAMETERS

pPalInfo - input/output

A pointer to a **PALINFO** structure (pg. 335) filled with information about this palette object.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to set information about a palette object. The `wpQueryPaletteInfo` method should be called first to retrieve the current settings of the palette.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wppalet.h

SEE ALSO

wpQueryPaletteInfo -330, wpSetupCell -333

NOTES

The xCursor and yCursor fields in the PALINFO structure cannot be changed.

● wpSetupCell WPPalette instance method

Sets the class-specific data on a cell of a palette object.

SYNTAX

BOOL wpSetupCell(**PVOID** *pCellData*, **ULONG** *cb*, **ULONG** *x*,
ULONG *y*)

PARAMETERS

pCellData - input

A pointer to the class-specific data to assign to this cell.

cb - input

The size of the *pCellData* buffer.

x - input

The column number of this cell.

y - input

The row number of this cell.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method when the object is initialized on each cell to add the class-specific data. Call wpQueryPaletteInfo to determine the number of rows and columns in this palette. A copy of the data will be made for the cell; therefore, it can be freed after this call.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wppalet.h

SEE ALSO

wpQueryPaletteInfo -330, wpSetup -63, wpSetupOnce -64

RESTRICTIONS/WARNINGS

- Call wpSetupCell only after the palette has been completely initialized. For OS/2 version 2.1, Override wpSetup to set up the cells but ensure wpSetup is being called on a new object and not from WinSetObjectData. For OS/2 Warp or later, set up the cells from the wpSetupOnce override.

Class Methods

wpclsQueryEditString returns the string used for the edit pushbutton in the palette view (pg. 334).

● **wpclsQueryEditString WPPalette class method**

Returns the string used for the edit pushbutton in the palette view.

SYNTAX

PSZ wpclsQueryEditString()

PARAMETERS

none

RETURNS

(nonzero value)—A pointer to the edit string for this class.
NULL—This class does not have an edit string defined.

HOW TO CALL

Call this method at any time to query the edit string for the class.

HOWTO OVERRIDE

Override this method to return the edit string for the class. The system will automatically resize the edit button to fit the length of the string. This string will be accessed by the system each time a palette window is opened so the memory for the string must be valid for as long as the class is loaded. A global variable can be used for this purpose.

OTHER INFO

Include files: wppalet.h

Palette Objects: Structures

CELL

(4 bytes + class-specific data)

cbData (ULONG)

0

The size of the cell data.

Note: The CELL structure is immediately followed by class-specific data.

Used by: wpDragCell -327, wpEditCell -328, wpPaintCell -329, wpRedrawCell -330, wpSelectCell -331

PALINFO

(32 bytes)

xCell Count (ULONG)

0

The number of columns of cells.

yCellCount (ULONG)

4

The number of rows of cells.

xCursor (ULONG)

8

The x position of the cursor. This value cannot be changed.

yCursor (ULONG)

12

The y position of the cursor. This value cannot be changed.

xCellWidth (ULONG)

16

The width of each cell in pixels.

yCellHeight (ULONG)

20

The height of each cell in pixels.

xGap (ULONG)	24
The gap, in pixels, between each column of cells.	
yGap (ULONG)	28
The gap, in pixels, between each row of cells.	
Used by: wpQueryPaletteInfo -330, wpSetPaletteInfo -332	

RECTL**(16 bytes)**

xLeft (ULONG)	0
The x coordinate for the lower left point of the rectangle.	
yBottom (ULONG)	4
The y coordinate for the lower left point of the rectangle.	
xRight (ULONG)	8
The x coordinate for the upper right point of the rectangle.	
yTop (ULONG)	12
The y coordinate for the upper right point of the rectangle.	
Used by: wpPaintCell -329	

●18

Help

Workplace Shell provides help primarily by creating a global help instance for all objects. Application help files are appended when `wpDisplayHelp` is called. This help instance is used to provide general object help, help for pop-up menu items, or to display individual help panels at any time.

Settings notebooks each have their own help instances which are created just after all classes have inserted their pages. The `PAGEINFO` structure passed to `wpInsertSettingsPage` contains fields for the general help panel, help library, and help table for each page.

Restrictions and Warnings

- After calling `wpDisplayHelp`, the application's help library is included in the Workplace help instance and, therefore, will be in use. If you need to delete or replace this file, press F1 when an object from another class is selected and your library file will be unloaded.

Instance Methods

wpDisplayHelp displays a help panel from the specified help library (pg. 338).

wpMenuItemHelpSelected notifies an object that the user summoned help for a pop-up menu item (pg. 339).

wpQueryDefaultHelp returns the general help panel for an object (pg. 340).

wpSetDefaultHelp sets the general help panel for an object (pg. 341).

● **wpDisplayHelp** **WPObject instance method**

Displays a help panel from the specified help library.

SYNTAX

BOOL `wpDisplayHelp(ULONG HelpPanelId, PSZ HelpLibrary)`

PARAMETERS

HelpPanelId - input

The resource ID for the help panel to display.

HelpLibrary - input

A pointer to a string containing the name of the help file where *HelpPanelId* is contained. The file name does not need to be fully qualified if the help file is in the HELP path specified in CONFIG.SYS.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time to display a help panel for an object. Use this method to display the help panels for menu items in the `wpMenuItemHelpSelected` override. The system will call this method using the values returned from `wpQueryDefaultHelp` to display general help for an object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wobject.h`

SEE ALSO

`wpclsQueryDefaultHelp` -342, `wpMenuItemHelpSelected` -339, `wpQueryDefaultHelp` -340, `wpSetDefaultHelp` -341

● **`wpMenuItemHelpSelected`** **WObject instance method**

Notifies an object that the user summoned help for a pop-up menu item.

SYNTAX

BOOL `wpMenuItemHelpSelected`(**ULONG** *MenuId*)

PARAMETERS

MenuId - input

The ID of the menu item for which help was requested.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method is generally only called by the system when the user pressed F1 while a pop-up menu item has the focus.

HOW TO OVERRIDE

Override this method to provide help for menu items added by your class. If *MenuId* is not one of your menu items, call the parent method.

Otherwise, call `wpDisplayHelp` to display the help panel. (See sample below.)

OTHER INFO

Include files: `wpobject.h`

SEE ALSO

`wpclsQueryDefaultHelp` -342, `wpDisplayHelp` -338,
`wpQueryDefaultHelp` -340, `wpSetDefaultHelp` -341

SAMPLE CODE

This sample demonstrates how to override `wpMenuItemHelpSelected`.

The following is from the `.h` file:

```
#define IDM_MYMENUITEM WPMENUID_USER
#define RES_MYMENUITEM 1000
```

The following is from the `.c` file:

```
SOM_Scope BOOL SOMLINK mcls_wpMenuItemHelpSelected(MyClass *somSelf,
                                                    ULONG MenuId)
{
    /* Check if the menu id is one added by this class. */
    if (MenuId == IDM_MYMENUITEM)
        return (_wpDisplayHelp(somSelf, RES_MYMENUITEM,
                               "MYCLASS.HLP"));
    else
        return (parent_wpMenuItemHelpSelected(somSelf, MenuId));
}
```

● **`wpQueryDefaultHelp` WPObject instance method**

Returns the general help panel for an object.

SYNTAX

BOOL `wpQueryDefaultHelp`(**PULONG** *pHelpPanelId*, **PSZ** *HelpLibrary*)

PARAMETERS

pHelpPanelId - output

A pointer to a **ULONG** which, upon return, will contain the resource ID of the general help panel for this object.

HelpLibrary - input/output

A pointer to a string buffer which, upon return, will contain the help library name for this object. This buffer must be at least the size of CCHMAXPATH.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to determine the general help panel and library for an object. Pass *pHelpPanelId* and *HelpLibrary* to `wpDisplayHelp` to display the help panel. By default, the object will return the values returned from `wpclsQueryDefaultHelp`.

HOW TO OVERRIDE

This method is generally not overridden. Call `wpSetDefaultHelp` to change the default help panel for an object.

OTHER INFO

Include files: `wpobject.h` and `bsedos.h`

SEE ALSO

`wpclsQueryDefaultHelp` -342, `wpDisplayHelp` -338,
`wpSetDefaultHelp` -341

● **wpSetDefaultHelp** **WPObj** instance method

Sets the general help panel for an object.

SYNTAX

BOOL `wpSetDefaultHelp`(**ULONG** *HelpPanelId*, **PSZ** *HelpLibrary*)

PARAMETERS

HelpPanelId - input

The resource ID of the general help panel for this object.

HelpLibrary - input

A pointer to a string containing the help library name for this object. The size of this string must be less than or equal to CCHMAXPATH. If

the help file is in the HELP path in CONFIG.SYS, *HelpLibrary* does not need to be fully qualified.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to set the general help panel and library for an object. By default, the object will use the values returned from the `wpclsQueryDefaultHelp` override.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpobject.h` and `bsedos.h`

SEE ALSO

`wpclsQueryDefaultHelp` -342, `wpDisplayHelp` -338,
`wpQueryDefaultHelp` -340

Class Methods

`wpclsQueryDefaultHelp` returns the general help panel for a class (pg. 342).

● `wpclsQueryDefaultHelp` **WPObj class method**

Returns the default help panel for a class.

SYNTAX

BOOL `wpclsQueryDefaultHelp`(**PULONG** *pHelpPanelId*, **PSZ** *HelpLibrary*)

PARAMETERS

pHelpPanelId - output

A pointer to a ULONG which, upon return, will contain the resource ID of the general help panel for this object.

HelpLibrary - input/output

A pointer to a string buffer which, upon return, will contain the help library name for this object. This buffer must be at least the size of CCHMAXPATH.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to determine the class default general help panel.

HOW TO OVERRIDE

Override this method to return the general help panel for this class. All instances will use this panel as their general help unless `wpSetDefaultHelp` is called. When the system calls this method, the size of the *HelpLibrary* buffer is CCHMAXPATH.

OTHER INFO

Include files: `wobject.h` and `bsedos.h`

SEE ALSO

`wpDisplayHelp` -338, `wpQueryDefaultHelp` -340, `wpSetDefaultHelp` -341

●19

Printing

The Printer Object

When a print queue is created in the system, the system creates a printer object on the desktop to represent the queue. Printer objects are descendants of WPPrinter and provide the user interface for the print subsystem. Users can change settings on the print queues using the settings notebook, and can print objects by dragging and dropping them on the printer objects.

Printing Workplace Objects

Any Workplace class can specify print behavior for its objects. For an object to be printable, its style must not contain `OBJSTYLE_NO-PRINT`. An entire class of objects can be defined as printable by removing the `CLSSTYLE_NEVERPRINT` flag from the class default style.

When the user selects Print from the pop-up menu or drags the object to the printer, the `wpPrintObject` method is called. It is then up to the object to print the data. The only default Workplace class that handles the `wpPrintObject` method is `WPDataFile`. A `WPDataFile` subclass can change the print behavior by overriding one of the `WPPrintXxxFile` methods.

Restrictions and Warnings

- Print objects can only be created by the SplXxx APIs in OS/2 versions before Warp.

Instance Methods

wpDeleteAllJobs deletes all the jobs in a printer (pg. 346).

wpHoldPrinter holds the queue of a printer (pg. 346).

wpJobAdded (Warp) notifies a printer that a job has been added to the queue (pg. 347).

wpJobChanged (Warp) notifies a printer that a job in the queue has been changed (pg. 348).

wpJobDeleted (Warp) notifies a printer that a job has been removed from the queue (pg. 348).

wpPrintMetaFile prints a file in metafile format (pg. 349).

wpPrintObject notifies an object that a print request has been made by the user (pg. 350).

wpPrintPifFile prints a file in PIF format (pg. 358).

wpPrintPlainTextFile prints a file in plain text format (pg. 359).

wpPrintPrinterSpecificFile prints a file in printer-specific format (pg. 360).

wpPrintUnknownFile prompts the user for the format of a file and then calls the appropriate print method (pg. 361).

wpReleasePrinter releases the queue of a printer (pg. 362).

wpQueryComputerName returns the computer name of a printer (pg. 363).

wpQueryPrinterName returns the printer name for a printer (pg. 363).

wpQueryQueueOptions (Warp) returns the queue options of a printer (pg. 364).

wpQueryRemoteOptions (Warp) returns the remote options of a printer (pg. 365).

wpSetDefaultPrinter sets a printer object as the default printer (pg. 366).

wpSetQueueOptions (Warp) sets the queue options of a printer (pg. 367).

wpSetRemoteOptions (Warp) sets the remote options of a printer (pg. 368).

- **wpDeleteAllJobs** **WPPrinter instance method**

Deletes all the jobs in a printer object.

SYNTAX

BOOL wpDeleteAllJobs()

PARAMETERS

none

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time to delete the pending jobs from the queue of a printer object. The user must be logged on as a network administrator for this method to have an effect on a network printer.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpprint.h

SEE ALSO

wpHoldPrinter -346, wpReleasePrinter -362

- **wpHoldPrinter** **WPPrinter instance method**

Holds the queue of a printer object.

SYNTAX

BOOL wpHoldPrinter()

PARAMETERS

none

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time on a printer object to hold the queue. The user must be logged on as a network administrator for this method to have an effect on a network printer.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpprint.h`

SEE ALSO

`wpDeleteAllJobs` -346, `wpReleasePrinter` -362

● wpJobAdded WPPrinter instance method (Warp only)

Notifies a printer that a job has been added to the queue.

SYNTAX

BOOL wpJobAdded(**ULONG** *ulJobId*)

PARAMETERS

ulJobId - input

The ID of the job added to the queue.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method is generally only called by the system.

HOW TO OVERRIDE

Override this method for notification when a job is added to the print queue. You must call the parent method.

OTHER INFO

Include files: `wpprint.h`

SEE ALSO

`wpJobChanged` -348, `wpJobDeleted` -348

● wpJobChanged WPPrinter instance method (Warp only)

Notifies a printer that a job in the queue has been changed.

SYNTAX

BOOL `wpJobChanged(ULONG ulJobId, ULONG ulReserved)`

PARAMETERS

ulJobId - input

The ID of the job that has changed.

ulReserved - reserved

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method is generally only called by the system.

HOW TO OVERRIDE

Override this method for notification when a job in the print queue has changed. You must call the parent method.

OTHER INFO

Include files: `wpprint.h`

SEE ALSO

`wpJobAdded` -347, `wpJobDeleted` -348

● wpJobDeleted WPPrinter instance method (Warp only)

Notifies a printer that a job has been removed from the queue.

SYNTAX

BOOL wpJobDeleted(**ULONG** *ulJobId*)

PARAMETERS

ulJobId - input

The ID of the job removed from the queue.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method is generally only called by the system.

HOW TO OVERRIDE

Override this method for notification when a job is removed from the print queue. You must call the parent method.

OTHER INFO

Include files: wpprint.h

SEE ALSO

wpJobAdded -347, wpJobChanged -348

● wpPrintMetaFile WPDataFile instance method

Prints a file in metafile format.

SYNTAX

BOOL wpPrintMetaFile(**PPRINTDEST** *pPrintDest*)

PARAMETERS

pPrintDest - input

A pointer to a PRINTDEST structure (pg. 369) containing the information needed to open the device context for the desired printer.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time to print a data file object in metafile format. The `WPDataFile` class handles this method by invoking a function in the Picture Viewer (`PICVIEW.EXE`) to print the metafile.

HOW TO OVERRIDE

This method can be overridden to replace the print behavior of metafiles. When the user requests to print a metafile, the system will invoke `wpPrintObject` on the file. The `WPDataFile` class will respond by calling this method.

OTHER INFO

Include files: `wpdataf.h`

SEE ALSO

`wpPrintObject` -350, `wpPrintPiffFile` -358, `wpPrintPlainTextFile` -359, `wpPrintPrinterSpecificFile` -360, `wpPrintUnknownFile` -361

● **wpPrintObject** **WPObj** instance method

Notifies an object that a print request has been made by the user.

SYNTAX

BOOL `wpPrintObject`(**PPRINTDEST** *pPrintDest*, **ULONG** *ulReserved*)

PARAMETERS

pPrintDest - input

A pointer to a `PRINTDEST` structure (pg. 369) containing the information needed to open the device context for the desired printer.

ulReserved - reserved

Set this parameter to 0.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred or printing is not supported.

HOW TO CALL

This method can be called at any time to print an object that supports printing. To determine if an object can be printed, call `wpQueryStyle` and check that the `OBJSTYLE_NOPRINT` flag is not returned.

HOW TO OVERRIDE

This method can be overridden to provide printing for an object. If you are replacing the print functionality of the object or the parent class does not support printing, do not call the parent method. `WPDataFile` is the only predefined `Workplace` class that supports printing.

The system will call this method from the user interface thread of Workplace Shell. `wpPrintObject` processing should be done on a separate thread to enable the user to continue to interact with the system while the object is printing. Be sure to make a copy of the `pPrintDest` parameter to pass to the thread (see sample below).

OTHER INFO

```
Include files:  wproject.h
```

SEE ALSO

wpPrintMetaFile -349, wpPrintPiffFile -358, wpPrintPlainTextFile -359,
wpPrintPrinterSpecificFile -360, wpPrintUnknownFile -361

SAMPLE CODE

This sample demonstrates how to provide printing for a `WPFolder` subclass. When `wpPrintObject` is called, a separate thread will print the contents of the folder. It is beyond the scope of this book to delve into the details of PM printing, therefore, the `PrintFolder` function is a crude example that does not support font selection or other important aspects of printing.

The following is from the .h file:

```

/* structure for the thread parameter */
typedef struct _THREADDATA
{
    WPFolder *Folder;
    PPRINTDEST pPrintDest;
} THREADDATA, *PTHREADDATA;

VOID PrintFolder(PTHREADDATA pData);
PPRINTDEST CreateDupPrintDest(PPRINTDEST pPrintDest);
VOID FreePrintDest(PPRINTDEST pPrintDest);

```

```

{
    PPRINTDEST pDupPrintDest;
    PTHREADDATA pData;
    TID      tid;

    /* Make a copy of the PRINTDEST structure so the thread can use it
       after this method returns.
    */
    pDupPrintDest = CreateDupPrintDest(pPrintDest);
    if (pDupPrintDest)
    {
        pData = malloc(sizeof(THREADDATA)); /* Create the thread
                                              parameter */

        if (pData)
        {
            pData->Folder = somSelf;
            pData->pPrintDest = pDupPrintDest;

            /* Lock this folder so it doesn't go to sleep while
               printing */
            _wpLockObject(somSelf);
            if (DosCreateThread(&tid, (PFNTHREAD) PrintFolder,
                               (ULONG) pData,
                               0, 0x10000))
            {
                free(pData);
                FreePrintDest(pDupPrintDest);
                _wpUnlockObject(somSelf);
            }
            else
                return(TRUE);
        } /* endif pData */
        else
            FreePrintDest(pDupPrintDest);
    }
    return (FALSE);
}

#undef SOM_CurrentClass
#define SOM_CurrentClass SOMMeta

/* Override wpclsQueryStyle to remove the CLSSTYLE_NEVERPRINT flag */

```

```

SOM_Scope ULONG SOMLINK mfdrM_wpclsQueryStyle(M_MyFolder *somSelf)
{
    return (parent_wpclsQueryStyle(somSelf) &~ CLSSTYLE_NEVERPRINT);
}

PPRINTDEST CreateDupPrintDest(PPRINTDEST pPrintDest)
{
    PPRINTDEST pDupPrintDest;
    PSZ        *apdopData, pszValue;
    ULONG      ulIndex, ulSize;
    PDRIVDATA  pDrivData, pOldDrivData;
    BOOL       bFreePrintDest = FALSE;

    /* Allocate a new PRINTDEST structure */
    pDupPrintDest = malloc(sizeof(PRINTDEST));
    if (!pDupPrintDest)
        return(NULL);

    memset(pDupPrintDest, 0, sizeof(PRINTDEST));
    /* Copy over the values from pPrintDest and make new string
     * pointers for string fields.
     */
    pDupPrintDest->cb = sizeof(PRINTDEST);
    pDupPrintDest->lType = pPrintDest->lType;
    pDupPrintDest->pszToken = pPrintDest->pszToken
        ? strdup(pPrintDest->pszToken)
        : NULL;
    pDupPrintDest->lCount = pPrintDest->lCount;
    pDupPrintDest->fl = pPrintDest->fl;
    pDupPrintDest->pszPrinter = pPrintDest->pszPrinter
        ? strdup(pPrintDest->pszPrinter)
        : NULL;

    /* Loop through pPrintDest->lCount and create a new pdopData
     array. */
    ulSize = pPrintDest->lCount * sizeof(PSZ);
    apdopData = malloc(ulSize);
    if (apdopData)
    {
        memset(apdopData, 0, ulSize);
        for (ulIndex = 0; ulIndex < pPrintDest->lCount; ulIndex++)
        {
            /* If this index is not DRIVER_DATA, the data is a string so
             * just create a new string buffer with strdup.

```

```

        */
        if (ulIndex != DRIVER_DATA)
        {
            pszValue = pPrintDest->pdopData[ulIndex];
            apdopData[ulIndex] = pszValue ? strdup(pszValue) : NULL;
        }
        else
        {
            /* This is the driver data. Copy the structure over */
            pOldDrivData = (PDRIVDATA)pPrintDest->pdopData[ulIndex];
            pDrivData = malloc(pOldDrivData->cb);
            if (pDrivData)
            {
                memcpy(pDrivData, pOldDrivData, pOldDrivData->cb);
                apdopData[ulIndex] = (PSZ)pDrivData;
            }
            else
            {
                bFreePrintDest = TRUE;
                break;
            }
        }
    }
    pDupPrintDest->pdopData = apdopData;
}
else
    bFreePrintDest = TRUE;

if (bFreePrintDest)
{
    /* An error occurred somewhere above. Free the pDupPrintDest
       buffer. */
    FreePrintDest(pDupPrintDest);
    return(NULL);
}
else
    return(pDupPrintDest);
}

/* FreePrintDest frees all the data associated with a PRINTDEST
structure */
VOID FreePrintDest(PPRINTDEST pPrintDest)
{

```



```
        ULONG ulIndex;

        /* Free the strings. */
        free(pPrintDest->pszToken);

        free(pPrintDest->pszPrinter);

        /* Free the elements in the pdopData array */
        for (ulIndex = 0; ulIndex < pPrintDest->lCount; ulIndex++ )
        {
            free(pPrintDest->pdopData[ulIndex]);
        }

        /* Free the array itself */
        free(pPrintDest->pdopData);

        free(pPrintDest);
    }

    /* PrintFolder is the thread procedure that actually prints out the
     * contents of the folder.
     */
    VOID PrintFolder(PTHREADDATA pData)
    {
        HDC hdc;
        HPS hps;
        SIZEL sizel;
        CHAR szDocTitle[256];
        PSZ pszFileName, pszTitle;
        ULONG ulSize;
        LONG lCharHeight;
        SOMany *Object, *NextObj;
        POINTL point;
        HMQ hmq = 0;
        HAB hab = 0;
        FONTMETRICS fm;

        /* This thread needs a message queue to do printing */
        hab = WinInitialize(0);
        if (!hab)
            return;
        hmq = WinCreateMsgQueue(hab, 0);
        if (hmq)
```



```
hdc = DevOpenDC(hab, pData->pPrintDest->lType,
               pData->pPrintDest->pszToken,
               pData->pPrintDest->lCount,
               pData->pPrintDest->pDopData, NULLHANDLE);
if (hdc)
{
    /* Get the width and height of the paper. */
    DevQueryCaps(hdc, CAPS_WIDTH, 2L, (PLONG) &szel);

    /* Create a presentation space to draw in. */
    hps = GpiCreatePS(hab, hdc, &szel, PU_TWIPS | GPIT_MICRO |
                     GPIF_DEFAULT | GPIA_ASSOC);
    /* Query the character height */
    GpiQueryFontMetrics(hps, sizeof(FONTMETRICS), &fm );
    lCharHeight = fm.lMaxBaselineExt + fm.lExternalLeading;

    /* Give the printer the name of this new job. */
    pszTitle = _wpQueryTitle(pData->Folder);
    sprintf(szDocTitle, "Folder Contents for %s", pszTitle);
    DevEscape(hdc, DEVEESC_STARTDOC, sizeof(szDocTitle), szDocTitle,
              NULL, NULL);

    /* Draw the title at the top of the page */
    point.x = 10;
    point.y = szel.cy - lCharHeight;
    GpiSetCurrentPosition(hps, &point);
    GpiCharString(hps, strlen(szDocTitle), szDocTitle);
    point.y -= lCharHeight * 2;

    /* Populate the folder and loop through the contents */
    _wpQueryRealName(pData->Folder, NULL, &ulSize, TRUE);
    if (ulSize)
    {
        pszFileName = malloc(ulSize);
        _wpQueryRealName(pData->Folder, pszFileName, &ulSize, TRUE);
        _wpPopulate(pData->Folder, 0, pszFileName, FALSE);
        free(pszFileName);

        Object = _wpQueryContent(pData->Folder, NULL, QC_First);
        while (Object)
        {
            /* Print the title of this object */
            pszTitle = _wpQueryTitle(Object);
```

```

        GpiSetCurrentPosition(hps, &point);
        GpiCharString(hps, strlen(pszTitle), pszTitle);

        point.y -= lCharHeight;

        /* Get the next object THEN unlock this one */
        NextObj = _wpQueryContent(pData->Folder, Object,
                                QC_Next);
        _wpUnlockObject(Object);
        Object = NextObj;

        /* If we're at the bottom of the page, eject and start a
        new one. */
        if ((point.y <= lCharHeight*2) && Object)
        {
            DevEscape(hdc, DEVEESC_NEWFRAME, 0L, NULL, NULL, NULL );
            point.y = sizel.cy - lCharHeight;
        }
    }
    DevEscape(hdc, DEVEESC_ENDDOC, 0, NULL, NULL, NULL);
    GpiDestroyPS(hps);
    DevCloseDC(hdc);
}

_wpUnlockObject(pData->Folder);

FreePrintDest(pData->pPrintDest);
free(pData);

if (hmq)
    WinDestroyMsgQueue(hmq);

if (hab)
    WinTerminate(hab);
}

```

● **wpPrintPifFile** **WPDataFile instance method**

Prints a file in PIF format.

SYNTAX

BOOL wpPrintPifFile(**PPRINTDEST** *pPrintDest*)

PARAMETERS

pPrintDest - input

A pointer to a PRINTDEST structure (pg. 369) containing the information needed to open the device context for the desired printer.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time to print a PIF file object. The WPDataFile class handles this method by invoking a function in the Picture Viewer (PICVIEW.EXE) to print the PIF file.

HOW TO OVERRIDE

This method can be overridden to replace the print behavior of PIF files. When the user requests to print a PIF file, the system will invoke wpPrintObject on the file. The WPDataFile class will respond by calling this method.

OTHER INFO

Include files: wpdataf.h

SEE ALSO

wpPrintMetaFile -349, wpPrintObject -350, wpPrintPlainTextFile -359, wpPrintPrinterSpecificFile -360, wpPrintUnknownFile -361

● wpPrintPlainTextFile WPDataFile instance method

Prints a file in plain text format.

SYNTAX

BOOL wpPrintPlainTextFile(**PPRINTDEST** *pPrintDest*)

PARAMETERS

pPrintDest - input

A pointer to a PRINTDEST structure (pg. 369) containing the information needed to open the device context for the desired printer.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time to print a plain text file object.

HOW TO OVERRIDE

This method can be overridden to replace the print behavior of plain text files. When the user requests to print a file and selects Plain text from the type selection dialog, the WPDataFile class calls this method.

OTHER INFO

Include files: wpdataf.h

SEE ALSO

wpPrintMetaFile -349, wpPrintObject -350, wpPrintPifFile -358, wpPrintPrinterSpecificFile -360, wpPrintUnknownFile -361

● wpPrintPrinterSpecificFile WPDataFile instance method

Prints a file in printer-specific format.

SYNTAX

BOOL wpPrintPrinterSpecificFile(**PPRINTDEST** *pPrintDest*)

PARAMETERS

pPrintDest - input

A pointer to a PRINTDEST structure (pg. 369) containing the information needed to open the device context for the desired printer.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time to print a file with printer-specific data.

HOW TO OVERRIDE

This method can be overridden to replace the print behavior of printer-specific data. When the user requests to print a file and selects Printer-specific from the type selection dialog, the `WPDataFile` class calls this method.

OTHER INFO

Include files: `wpdataf.h`

SEE ALSO

`wpPrintMetaFile` -349, `wpPrintObject` -350, `wpPrintPiffFile` -358, `wpPrintPlainTextFile` -359, `wpPrintUnknownFile` -361

● `wpPrintUnknownFile` **WPDataFile** instance method

Prompts the user for the format of a file and then calls the appropriate print method.

SYNTAX

BOOL `wpPrintUnknownFile`(**PPRINTDEST** *pPrintDest*)

PARAMETERS

pPrintDest - input

A pointer to a `PRINTDEST` structure (pg. 361) containing the information needed to open the device context for the desired printer.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time to print a file whose format is unknown. The type selection dialog will be displayed to allow the user to choose between Plain text and Printer-specific formats.

HOW TO OVERRIDE

This method can be overridden to replace the behavior of the type selection dialog.

OTHER INFO

Include files: `wpdataf.h`

SEE ALSO

`wpPrintMetaFile` -349, `wpPrintObject` -350, `wpPrintPiffFile` -358, `wpPrintPlainTextFile` -359, `wpPrintPrinterSpecificFile` -360

● wpReleasePrinter WPPrinter instance method

Releases the queue of a printer object.

SYNTAX

BOOL `wpReleasePrinter()`

PARAMETERS

none

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time on a printer object to release the queue. The user must be logged on as a network administrator for this method to have an effect on a network printer.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpprint.h`

SEE ALSO

`wpDeleteAllJobs` -346, `wpHoldPrinter` -346

● wpQueryComputerName WPPrinter instance method

Returns the computer name of a printer object.

SYNTAX

ULONG wpQueryComputerName(**PSZ** *pszComputerName*)

PARAMETERS

pszComputerName - input/output

A pointer to a string buffer which, upon return, will contain the computer name for this print device if the printer resides on the network. The size of this buffer should be CCHMAXPATH.

RETURNS

0—An error occurred.

1—The printer is local.

2—The printer is on the network. *pszComputerName* contains the network computer name.

HOW TO CALL

This method can be called at any time on a printer object to query the computer name.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpprint.h

● wpQueryPrinterName WPPrinter instance method

Returns the physical printer name of a printer object.

SYNTAX

BOOL wpQueryPrinterName(**PSZ** *pszPrinterName*)

PARAMETERS

pszPrinterName - input/output

A pointer to a string buffer which, upon return, will contain the physical printer name for this print device. The size of this buffer should be CCHMAXPATH.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time on a printer object to query the physical printer name. Note that this name is different from the title of the printer object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpprint.h`

●	wpQueryQueueOptions	WPPrinter instance method (Warp only)
---	----------------------------	--

Returns the queue options of a printer.

SYNTAX

ULONG wpQueryQueueOptions()

PARAMETERS

(none)

RETURNS

The queue options for this printer which can contain the following flags ORed together:

Define		Description
PO_PRINTERSPECIFICFORMAT	0×01	Spool print jobs in PM_Q_RAW format.
PO_PRINTWHILESPooling	0×02	Printing is enabled while a job is spooling.

PO_APPDEFAULT	0×04	This is the default printer object.
PO_JOBIALOGBEFOREPRINT	0×10000	Display the job properties dialog when a job is printed.

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpprint.h

SEE ALSO

wpSetQueueOptions -367

● wpQueryRemoteOptions WPPrinter instance method (Warp only)

Returns the remote options of a printer.

SYNTAX

BOOL wpQueryRemoteOptions(**PULONG** *pulRefreshInterval*,
 PULONG *pflAllJobs*)

PARAMETERS

pulRefreshInterval - output

A pointer to the interval, in seconds, between refreshes of the printer views.

pflAllJobs - output

TRUE—Display all the jobs in the network queue.

FALSE—Display only the jobs for the current user in the network queue.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpprint.h`

SEE ALSO

`wpSetRemoteOptions` -368

● wpSetDefaultPrinter WPPrinter instance method

Sets a printer object as the default printer.

SYNTAX

BOOL wpSetDefaultPrinter()

PARAMETERS

none

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

This method can be called at any time on a printer object set it as the default printer.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpprint.h`

● **wpSetQueueOptions** **WPPrinter instance method**
(Warp only)

Sets the queue options of a printer.

SYNTAX

BOOL wpSetQueueOptions(**ULONG** *ulOptions*)

PARAMETERS

ulOptions - input

The queue options for this printer which can contain the following flags ORed together:

Define		Description
PO_PRINTERSPECIFICFORMAT	0x01	Spool print jobs in PM_Q_RAW format.
PO_PRINTWHILESPooling	0x02	Printing is enabled while a job is spooling.
PO_APPDEFAULT	0x04	This is the default printer object.
PO_JOBDialogBeforePrint	0x10000	Display the job properties dialog when a job is printed.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpprint.h

SEE ALSO

wpQueryQueueOptions -364

Printing: Structures

PRINTDEST

(28 bytes)

cb (ULONG) 0

The size of the PRINTDEST structure, in bytes, including cb itself. This value does not include the length of the string fields or the pdopData.

lType (LONG) 4

The type of device context to specify in the call to DevOpenDC.

Constant	Description
OD_QUEUED	2 The output is to be queued.
OD_DIRECT	5 The output is not to be queued.

pszToken (PSZ) 8

The device information token to specify in the call to DevOpenDC. This is always “*”.

lCount (LONG) 12

The number of items in pdopData. At least the first four items must be specified for OD_QUEUED devices.

pdopData (PDEVOPENDATA) 16

An array of pointers to data in the format of the DEVOPENSTRUC structure. The size of this array is specified in lCount.

fl (ULONG) 20

The following flag may be specified:

Constant	Description
PD_JOB_PROPERTY	0×01 DevPostDeviceModes should be called with DPDM_POSTJOBPROP before calling DevOpenDC.

pszPrinter (PSZ) 24

The printer name to specify when calling DevPostDeviceModes.

Used by: wpPrintMetaFile -349, wpPrintObject -350, wpPrintPifFile -358, wpPrintPlainTextFile -359, wpPrintPrinterSpecificFile -360, wpPrintUnknownFile -361

DEVOPENSTRUC**(36 bytes)**

pszLogAddress (PSZ) 0

The logical address for the device. If the device is OD_DIRECT, this value must be set to the logical device address (e.g. "LPT1"). If the device is OD_QUEUEUED, this must be the name of the queue.

pszDriverName (PSZ) 4

The name of the device driver. This value is required.

pdriv (PDRIVDATA) 8

Device driver-specific data.

pszDataType (PSZ) 12

The type of data that is to be queued. Either "PM_Q_STD" for standard format or "PM_Q_RAW" for raw format.

pszComment (PSZ) 16

A description that can be displayed by the spooler to the user. This field is optional.

pszQueueProcName (PSZ) 20

The name of the queue processor for queued output. This field is optional.

pszQueueProcParams (PSZ) 24

The parameter string for the queue processor for queued output. This field is optional.

pszSpoolerParams (PSZ) 28

The parameter string for the spooler for queued output. The following two parameters can be specified and must be separated by a space:

FORM= form name, where form name is the name of the form. Multiple form names can be specified, separated by a comma. Form names should be enclosed by double quotes if they contain the characters ';' or '='.

PRTY= n where n is the priority in the range of 1 to 99. The default priority is 50.

pszNetworkParams (PSZ) 32

The parameter string for the network program for queued output. The following parameter is defined: (Other parameters may be defined by the network program)

USER = user name. If the user name is not specified, a null user identifier is used.

Used by: PRINTDEST structure

●20

Miscellaneous Methods

Finding Objects

Objects are easy to find if they have been assigned object IDs or if their persistent handles have been stored, but sometimes this information is not available. It is not always feasible to keep track of objects using their handles or IDs. WPObj provides class methods to allow applications to enumerate all the objects of any class or to find an object matching specific search criteria.

Error Handling

One of the disadvantages of object-oriented design is the difficulty in determining the cause of errors. If a method is called on an object and an error occurs many levels deep in the call chain, how does the caller know where the failure took place? Workplace provides methods for setting and querying the last error that occurred for an object or a

class. Instance methods should set errors using `wpSetError` and class methods should use `wpclsSetError`.

Restrictions and Warnings

- Finding objects usually takes a long time to process and, therefore, should be done from a separate thread.
- There are few situations when Workplace default classes set object or class errors. Some `WPFileSystem` methods set base file system errors on objects during copy or move actions, and the find methods set errors to indicate the status of the search. The error handling methods are mostly useful for defining new application errors.

Instance Methods

wpFindMinWindow (Warp) returns the minimized window object given a window frame handle (pg. 372).

wpModuleForClass (Warp) returns the name of the module in which a class is defined (pg. 373).

wpQueryError returns the last error for an object (pg. 374).

wpReplacementIsInEffect (Warp) returns whether the specified class has been replaced (pg. 375).

wpSetError sets the last error for an object (pg. 375).

-
- **wpFindMinWindow** **WPMinWinViewer**
instance method (Warp only)
-

Returns the minimized window object given a window frame handle.

SYNTAX

WPObj * wpFindMinWindow(**HWND** *hwndFrame*)

PARAMETERS

hwndFrame - input

The handle of a frame window.

RETURNS

(nonzero value)—The minimized window object that represents this frame window when it is minimized. Minimized windows are instances of the `WPMinWindow` class.

NULL—This frame has not been minimized.

HOW TO CALL

Call this method at any time on the Minimized Window Viewer `<WP_VIEWER>`.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpmwv.h`

●	wpModuleForClass	WPClassManager instance method (Warp only)
---	-------------------------	---

Returns the name of the module in which a class is defined.

SYNTAX

PSZ wpModuleForClass(**PSZ** *pszClass*)

PARAMETERS

pszClass - input

The name of the class.

RETURNS

(non-zero value)—The name of the module containing the specified class.

NULL—An error occurred.

HOW TO CALL

Call this method at any time on `SOMClassMgrObject` to determine the name of the .DLL where a class resides. This is useful when the module must be loaded to obtain resources for the class.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpclsmgr.h`

● wpQueryError WPObject instance method

Returns the last error for an object.

SYNTAX

`ULONG wpQueryError()`

PARAMETERS

none

RETURNS

The last error that occurred for this object.

HOW TO CALL

Call this method at any time to query the last error for an object. Call `wpSetError` to set the error for an object to zero before performing an action. Then, if the action fails, call `wpQueryError` to get the cause of the error.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpobject.h`

SEE ALSO

`wpclsQueryError` -384, `wpclsSetError` -385, `wpSetError` -375

NOTES

Most default Workplace methods do not call `wpSetError` when an error occurs. Therefore, `wpQueryError` cannot always be used to determine the cause of an error.

● **wpReplacementIsInEffect** **WPClassManager instance method (Warp only)**

Returns whether the specified class has been replaced.

SYNTAX

BOOL wpReplacementIsInEffect(**PSZ** *pszOldClass*, **PSZ** *pszNewClass*)

PARAMETERS

pszOldClass - input

The name of the original class.

pszNewClass - input

The name of the replacement class.

RETURNS

TRUE—*pszOldClass* has been replaced by *pszNewClass*.

FALSE—The specified replacement is not in effect.

HOW TO CALL

Call this method at any time on SOMClassMgrObject.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: wpclsmgr.h

SEE ALSO

WinReplaceObjectClass -19

● **wpSetError** **WPObject instance method**

Sets the last error for an object.

SYNTAX

ULONG wpSetError(**ULONG** *ulErrorId*)

PARAMETERS

ulErrorId - input

The error for the object. This can be any application-defined or system-defined error code.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to set the last error for an object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpobject.h`

SEE ALSO

`wpclsQueryError` -384, `wpclsSetError` -385, `wpQueryError` -374

Class Methods

wpclsFindObjectEnd ends a find operation (pg. 376).

wpclsFindObjectFirst begins a find operation and specifies the criteria for the search (pg. 377).

wpclsFindObjectNext continues a find operation (pg. 382).

wpclsFindOneObject (Warp) displays the find dialog (pg. 383).

wpclsQueryError returns the last error for a class (pg. 384).

wpclsSetError sets the last error for a class (pg. 385).

● **`wpclsFindObjectEnd` **WPObject class method****

Ends a find operation.

SYNTAX

BOOL wpclsFindObjectEnd(**HFIND** *hFind*)

PARAMETERS

hFind - input

The handle of the find operation to terminate.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

Possible returns from `_wpclsQueryError`

WPERR_INVALID_HFIND 0x1715

HOW TO CALL

Call this method to terminate a find operation that was initiated with `wpclsFindObjectFirst`.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpobject.h`

SEE ALSO

`wpclsFindObjectFirst` -377, `wpclsFindObjectNext` -382

● **wpclsFindObjectFirst** **WPObj class method**

Begins a find operation and specifies the criteria for the search.

SYNTAX

BOOL wpclsFindObjectFirst(**PCLASS** *pClassList*, **PHFIND** *phFind*, **PSZ** *pszTitle*, **WPFolder** **Folder*, **BOOL** *fSubfolders*, **PVOID** *pExtendedCriteria*, **POBJECT** *pBuffer*, **PULONG** *pCount*)

PARAMETERS

pClassList - input

A pointer to a NULL-terminated array of class object pointers.

phFind - input/output

A pointer to an **HFIND** which, upon return, will contain the handle for this find operation. Use the handle when calling **wpclsFindObjectNext** and **wpclsFindObjectEnd**.

pszTitle - input

A pointer to a string containing the title filter for objects. Only objects whose title matches this string will be returned. The character '*' can be used anywhere in this string as a wildcard. Specify "*" if no filter is desired.

Folder - input

The object pointer for the folder where the search will start. **wpclsQueryFolder** can be used to obtain a folder object pointer.

fSubFolders - input

TRUE—Search all subfolders of *Folder*.

FALSE—Do not search subfolders.

pExtendedCriteria - input

This parameter currently is not supported and should be set to **NULL**.

pBuffer - input/output

A pointer to a buffer which, upon return, will contain the pointers to the objects that were found.

pCount - input/output

A pointer to a **ULONG** containing the number of object pointers that will fit in *pBuffer*. Upon return, this parameter will contain the actual number of objects returned.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

Possible returns from **_wpclsQueryError**

WPERR_INVALID_FOLDER	0x1712
WPERR_BUFFER_OVERFLOW	0x1713
WPERR_OBJECT_NOT_FOUND	0x1714
WPERR_INVALID_HFIND	0x1715
WPERR_INVALID_COUNT	0x1716
WPERR_INVALID_BUFFER	0x1717

HOW TO CALL

Call this method to begin a find operation. If **FALSE** is returned, call **wpclsQueryError** to get the error code. If the error is **WPERR_BUFFER_OVERFLOW**, there are more objects for the system to return than will fit in *pBuffer*. Call **wpclsFindObjectNext** to get the remaining objects. See the sample below for more information.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpobject.h`

SEE ALSO

`wpcIsFindObjectEnd` -376, `wpcIsFindObjectNext` -377,
`wpcIsQueryFolder` -74

RESTRICTIONS/WARNINGS

- ❶ Each object that is returned from `wpcIsFindObjectFirst` or `wpcIsFindObjectNext` is locked once by the system. Call `wpUnlockObject` on each object when the pointer is no longer needed.
- ❷ Sometimes `wpcIsFindObjectFirst` or `wpcIsFindObjectNext` will set the last error to `WPERR_BUFFER_OVERFLOW` even when there are no more objects. Always check the *pCount* returned to see if zero objects were returned.

SAMPLE CODE

The following example demonstrates how to use `wpcIsFindObjectFirst`, `wpcIsFindObjectNext`, and `wpcIsFindObjectEnd` to search for printers. This function is passed to `DosCreateThread` and is executed on a separate thread.

```
VOID HoldPrinterObjects(VOID)
{
    CLASS  apClass[2];
    HFIND  hFind;
    SOMAny *Desktop,
    POBJECT pBuffer;
    BOOL   bReturn;
    ULONG  ulError = 0, ulCount = 10;
    HMQ    hmq = 0;
    HAB    hab = 0;

    hab = WinInitialize(0);
    if (!hab)
        return;
    /* Create a message queue since some Workplace methods might
       require one. */
```

```

hmq = WinCreateMsgQueue(hab, 0);
if (hmq)
    WinCancelShutdown(hmq, TRUE);
else
{
    WinTerminate(hab);
    return;
}
apClass[0] = _WPPrinter;           // Search for Printer objects
apClass[1] = NULL;                 // NULL terminate the array

/* Get the object pointer for the desktop. (no need to lock it
 * since the Desktop never goes to sleep)
 */
Desktop = _wpclsQueryFolder(_WPObject, "<WP_DESKTOP>", FALSE);
if (Desktop)
{
    /* Allocate a buffer that can hold 10 SOMany pointers */
    pBuffer = malloc(sizeof(OBJECT) * 10);
    if (pBuffer)
    {
        _wpclsSetError(_WPObject, 0);
        bReturn = _wpclsFindObjectFirst(_WPObject, apClass, &hFind,
                                         "",
                                         Desktop, TRUE, NULL, pBuffer,
                                         &ulCount);

        if (!bReturn)
            ulError = _wpclsQueryError(_WPObject);

        if (!ulError || ulError == WPERR_BUFFER_OVERFLOW)
        {
            ProcessBuffer(pBuffer, ulCount);
            /* Keep looping until there are no more objects */
            while (ulCount && ulError == WPERR_BUFFER_OVERFLOW)
            {
                ulError = 0;
                ulCount = 10;
                _wpclsSetError(_WPObject, 0);
                bReturn = _wpclsFindObjectNext(_WPObject, hFind,
                                                pBuffer, &ulCount);
            }
        }
    }
}

```

```
        if (!bReturn)
            ulError = _wpclsQueryError(_WPObject);

        if (!ulError || ulError == WPERR_BUFFER_OVERFLOW)
        {
            ProcessBuffer(pBuffer, ulCount);
        }
    }
    _wpclsFindObjectEnd(_WPObject, hFind);
}
free(pBuffer);
}
}

WinDestroyMsgQueue(hmq);

WinTerminate(hab);
}

/* ProcessBuffer processes each printer object in the buffer and
 * calls wpHoldPrinter to hold the printer's queue.
 */
VOID ProcessBuffer(SOMany **pBuffer, ULONG ulCount)
{
    ULONG i;
    PSZ pszTitle;
    CHAR szMessage[256];

    for (i = 0; i < ulCount; i++)
    {
        _wpHoldPrinter(pBuffer[i]);
        _wpUnlockObject(pBuffer[i]);
    }
}
```

● **wpclsFindObjectNext** **WPObj** class method

Continues a find operation.

SYNTAX

BOOL wpclsFindObjectNext(**HFIND** *hFind*, **POBJECT** *pBuffer*,
 PULONG *pCount*)

PARAMETERS

hFind - input

The find handle returned from wpclsFindObjectFirst.

pBuffer - input/output

A pointer to a buffer which, upon return, will contain the pointers to the objects that were found.

pCount - input/output

A pointer to a **ULONG** containing the number of object pointers that will fit in *pBuffer*. Upon return, this parameter will contain the actual number of objects returned.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

Possible returns from `_wpclsQueryError`

WPERR_BUFFER_OVERFLOW	0x1713
WPERR_OBJECT_NOT_FOUND	0x1714
WPERR_INVALID_HFIND	0x1715
WPERR_INVALID_COUNT	0x1716
WPERR_INVALID_BUFFER	0x1717

HOW TO CALL

Call this method to continue a find operation when the error from wpclsFindObjectFirst or wpclsFindObjectNext is **WPERR_BUFFER_OVERFLOW**; call it repeatedly until no more objects are found (**WPERR_OBJECT_NOT_FOUND** is returned). See the sample for wpclsFindObjectFirst for more information.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wobject.h`

SEE ALSO

`wpcsFindObjectEnd` -376, `wpcsFindObjectFirst` -377

RESTRICTIONS/WARNINGS

- ❶ Each object that is returned from `wpcsFindObjectFirst` or `wpcsFindObjectNext` is locked once by the system. Call `wpUnlockObject` on each object when the pointer is no longer needed.
- ❷ Sometimes `wpcsFindObjectFirst` or `wpcsFindObjectNext` will set the last error to `WPERR_BUFFER_OVERFLOW` even when there are no more objects. Always check the *pCount* returned to see if zero objects were returned.

❶ `wpcsFindObject` **WObject class method**

Displays the find dialog.

SYNTAX

WObject * `wpcsFindObject`(**HWND** *hwndOwner*, **PSZ** *pszFindParams*)

PARAMETERS

hwndOwner - input

The handle of the window that should be the owner of the find dialog.

pszFindParams - input

A setup string containing the search criteria. See `wpSetup` for more information about setup strings.

RETURNS

(non-zero value)—The object found by the system.

NULL—An error occurred or the object was not found.

HOW TO CALL

Call this method at any time to display the find dialog, enabling the user to search for an object.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpobject.h`

SEE ALSO

`wpclsFindObjectEnd` -376, `wpclsFindObjectFirst` -377,
`wpclsFindObjectNext` -382, `wpSetup` -63

● wpclsQueryError WPObject class method

Returns the last error for a class.

SYNTAX

ULONG `wpclsQueryError()`

PARAMETERS

none

RETURNS

The last error that occurred for this class.

HOW TO CALL

Call this method at any time to query the last error for a class. Call `wpclsSetError` to set the error for a class to zero before performing an action. Then, if the action fails, call `wpclsQueryError` to get the cause of the error.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpobject.h`

SEE ALSO

`wpclsSetError` -385, `wpQueryError` -374, `wpSetError` -375

NOTES

Most default Workplace class methods do not call `wpclsSetError` when an error occurs. Therefore, `wpclsQueryError` cannot always be used to determine the cause of an error.

● **wpclsSetError** **WPObject class method**

Sets the last error for a class.

SYNTAX

ULONG `wpclsSetError(ULONG ulErrorId)`

PARAMETERS

ulErrorId - input

The error for the class. This can be any application-defined or system-defined error code.

RETURNS

TRUE—The method completed successfully.

FALSE—An error occurred.

HOW TO CALL

Call this method at any time to set the last error for a class.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `wpobject.h`

SEE ALSO

`wpclsQueryError` -384, `wpQueryError` -374, `wpSetError` -375

21

System Object Model

It would take a separate book to document all of the information about IBM's System Object Model (SOM) but since this is a Workplace Shell programming reference, this chapter just highlights those aspects of SOM that are most relevant to WPS programmers.

Workplace Shell application developers must take special care in designing their SOM classes because they must coexist with other classes running in the same process. SOM provides the ability to replace entry points of some routines in order to change their behavior. The entry points for these replacement functions are stored in global variables which are common across the Workplace process. Workplace Shell already provides a replacement for `SOMError` to prevent the process from terminating. Workplace applications should use customized functions only for debugging purposes and not in retail versions of the code; otherwise, they will conflict with other applications in the Workplace process.

The shell also has programming interfaces that take the place of some basic SOM functionality. These include:

- `wpclsNew` is used instead of `somNew`.
- `wpDelete` is used instead of `somFree`.
- `wpInitData` and `wpUnInitData` should be overridden instead of `somInit` and `somUnInit`.
- `WinRegisterClass` is used instead of `somRegisterClass`.

This chapter does not include *all* SOM functions and macros but does contain those that are most useful for Workplace Shell programming. The methods are documented as they would appear in the SOM-STARs implementation of SOM 2.x header files in order to make them consistent with the SOM 1.0 header files.

Customization Functions

SOMInitModule initializes multiple classes packaged in a single .DLL (pg. 387).

SOMOutCharRoutine provides a replacement for character output (pg. 388).

● **SOMInitModule** **SOM customization function**

Initializes multiple classes packaged in a single .DLL.

SYNTAX

```
SOMEXTERN void SOMLINK SOMInitModule(long majorVersion,
                                         long minorVersion,
                                         zString className)
```

PARAMETERS

majorVersion - input

The major version requested by the client loading this class. In the case of Workplace Shell classes, Workplace itself is usually the client loading the class.

minorVersion - input

The minor version requested by the client loading this class.

className - input (**SOM 2.X only**)

The name of the requested class to load. If the class object for this class is not created, SOM will consider this load to have failed.

RETURNS

none

HOW TO USE

Replace this function by defining it in the .DLL containing the classes and exporting the function in the .DEF file. The SOMInitModule function should call <classname>NewClass for every class contained in the .DLL. This function is generally only called by the SOM class manager to load classes.

OTHER INFO

Include file: som.h

SEE ALSO

<class name>NewClass -394

RESTRICTIONS/WARNINGS

- This routine must use the _System linkage.

● SOMOutCharRoutine SOM customization function

Provides a replacement for character output.

SYNTAX

```
long MyOutCharRoutine (char c)
SOMOutCharRoutine = MyOutCharRoutine;
```

PARAMETERS

c - input

The character to write. The character output routine will be called once for every character to be written.

RETURNS

none

HOW TO USE

Replace this function by defining a character output routine using any function name. Then, during .DLL initialization, set the SOM global variable, SOMOutCharRoutine, to point to this function. From then

on, all Workplace Shell classes that use functions that call the `SOMOutCharRoutine` will use this customized function. The default character output routine writes the character to `stdout`.

OTHER INFO

Include file: `som.h`

SEE ALSO

<class name>`MethodDebug` -393, `somLPrintf` -390, `somPrintf` -391

NOTES

Only replace `SOMOutCharRoutine` in debug versions of your class implementation. If more than one Workplace application attempts to replace this function, they will conflict with each other and the most recent class to load will overwrite the `SOMOutCharRoutine` global variable.

RESTRICTIONS/WARNINGS

- This replacement routine must use the `_System` linkage.

Functions

somIdFromString returns the SOM ID for a text string (pg. 389).

somLPrintf prints a formatted string using `SOMOutCharRoutine` at the specified indentation level (pg. 390).

somPrintf prints a formatted string using `SOMOutCharRoutine` (pg. 391).

● **somIdFromString** **SOM function**

Returns the SOM ID for a text string.

SYNTAX

somId `somIdFromString` (**zString** *aString*)

PARAMETERS

aString - input

The string to convert to a SOM ID.

RETURNS

The SOM ID for the specified string.

HOW TO USE

Call this function at any time to get the SOM ID for a string. Many SOM methods have SOM IDs for parameters which are created using this function. It is the responsibility of the caller to free the ID using SOMFree.

OTHER INFO

Include file: som.h

SEE ALSO

somLocateClassFile -399

● somLPrintf SOM function

Prints a formatted string using SOMOutCharRoutine at the specified indentation level.

SYNTAX

int somLPrintf (**int** *level* , **zString** *fmt* , *args* ...)

PARAMETERS

level - input

The indentation level at which to place the string. $2 \times level$ spaces will be written before the string.

fmt - input

The format string to print.

args - input

Arguments to substitute in the format string.

RETURNS

The number of characters printed.

HOWTO USE

Call this function at any time to write a string. SOMOutCharRoutine will be used to output characters. The default destination is stdout but this can be customized. somLPrintf is similar to the C function printf, except it allows indentation levels and uses SOMOutCharRoutine instead of stdout.

OTHER INFO

Include files: somapi.h (SOM 2.X) or som.h (SOM 1.0)

SEE ALSO

SOMOutCharRoutine -388, somPrintf -391

● **somPrintf** **SOM function**

Prints a formatted string using SOMOutCharRoutine.

SYNTAX

int somPrintf (**zString** *fmt* , *args* ...)

PARAMETERS

fmt - input

The format string to print.

args - input

Arguments to substitute in the format string.

RETURNS

The number of characters printed.

HOWTO USE

Call this function at any time to write a string. SOMOutCharRoutine will be used to output characters. The default destination is stdout but this can be customized. somPrintf is similar to the C function printf, except it uses SOMOutCharRoutine instead of stdout.

OTHER INFO

Include files: somapi.h (SOM 2.X) or som.h (SOM 1.0)

SEE ALSO

SOMOutCharRoutine -388, somLPrintf -390

Generated Macros and Functions

<class name>GetData returns a pointer to the instance data for an object (pg. 392).

<class name>MethodDebug prints a debug message for a method using somPrintf (pg. 393).

<classname>NewClass creates a class object for <classname> (pg. 394).

_<class name> references to the class object representing <class name> (pg. 395).

● **<class name>GetData** **SOM generated macro**

Returns a pointer to the instance data for an object.

SYNTAX

<class name>Data * <class name>GetData (<class name> * *Object*)

For example, for the class MyClass, the macro MyClassGetData is defined, and takes a pointer of type MyClass* as a parameter and returns a pointer of type MyClassData*.

PARAMETERS

Object - input

The object for which the instance data is requested.

RETURNS

A pointer to the instance data for this object. This should be set to somThis to allow the _<variable name> macros to be used.

HOWTO USE

Call this macro to obtain a pointer to the instance data for an object. This macro is defined in the class' .ih file (for C) or .xih file (for C++).

SAMPLE CODE

The <class name>GetData macro should be called from within each method procedure that references instance variables. The following is an example of how to assign the somThis pointer from an instance method:

```
MyClassData *somThis = MyClassGetData(somSelf);
```

● <class name>MethodDebug SOM generated macro

Prints a debug message for a method using somPrintf.

SYNTAX

int <class name>MethodDebug(**PSZ** *pszClassName*, **PSZ** *pszMethodName*)

PARAMETERS

pszClassName - input

This is usually set to the name of the class; however, the caller can use any string because it will be passed directly to somPrintf.

pszMethodName - input

This is usually set to the name of the method; however, the caller can use any string because it will be passed directly to somPrintf.

RETURNS

The return from somPrintf.

HOW TO USE

Call this macro to print a debug statement. This macro resolves to the macro SOM_Trace, which prints a line using somPrintf. The line contains the source file name, the line number, *pszClassName*, and *pszMethodName*. In order to disable the effects of <classname>MethodDebug, include the class header files in the implementation source file, and then include the line:

```
#define <class name>MethodDebug(c,m) SOM_NoTrace(c,m)
```

SEE ALSO

somPrintf -391

SAMPLE CODE

The <class name>MethodDebug macro should be called from within each method procedure to debug the flow of execution for an object. The following is an example of how to use this macro:

```
MyClassMethodDebug("MyClass", "mcls_wpInitData");
```

● <class name>NewClass SOM generated function

Creates a class object for <class name>.

SYNTAX

<class name> * <class name>NewClass(**integer4** *MajorVersion*, **integer4** *MinorVersion*)

For example, the function prototype for MyClass is:

```
MyClass * MyClassNewClass (integer4 MajorVersion ,
                           integer4 MinorVersion)
```

PARAMETERS

MajorVersion - input

The requested major version number.

MinorVersion - input

The requested minor version number.

RETURNS

The class object pointer.

HOW TO USE

This function should only be called by Workplace Shell applications from within the SOMInitModule function to load multiple classes from one .DLL. If the class object has already been created, this function will return the existing class object pointer. The definition for this function is in the .ih (C) or the .xih (C++) file for the class.

SEE ALSO

SOMInitModule -387

● <class name> SOM generated macro

References the class object for <class name>.

SYNTAX

`_<class name>`

For example, the class object pointer for MyClass would be `_MyClass`.

HOWTO USE

Use this macro to reference the class object pointer for any class that is already loaded. Use `somFindClass` if the class may not be loaded. This macro is useful for passing class object pointers to class methods.

SOMObject Methods

somGetClass returns a class object pointer for the class of an object (pg. 395).

somGetClassName returns the name of the class of an object (pg. 396).

somIsA indicates whether an object is an instance of a class or its descendants (pg. 397).

somIsInstanceOf indicates if an object is an instance of a given class (pg. 397).

● somGetClass SOMObject method

Returns the class object pointer for the class of an object.

SYNTAX

`SOMClass * somGetClass ()`

PARAMETERS

none

RETURNS

The class object pointer for the object's class.

HOW TO CALL

Call this method on any object to obtain the class pointer for its class. This method is preferred over the `SOM_GetClass` macro.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `somobj.h`

SEE ALSO

`somGetClassName` -396

● somGetClassName SOMObject method

Returns the name of the class of an object.

SYNTAX

`zString somGetClassName ()`

PARAMETERS

none

RETURNS

The class name for the object's class.

HOW TO CALL

Call this method on any object to obtain the name of its class.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `somobj.h`

SEE ALSO

`somGetClass` -395

● somIsA SOMObject method

Indicates whether an object is an instance of a class or its descendants.

SYNTAX

int somIsA (**SOMClass** * *aClassObj*)

PARAMETERS

aClassObj - input

The class object to test the object against.

RETURNS

TRUE—The object is an instance of *aClassObj* or one of its descendants.

FALSE—The object is not one of these.

HOWTO CALL

Call this method on any object to test its class. This method is different from `somIsInstanceOf` in that it will also return **TRUE** if the class of the object is a descendant of *aClassObj*. `somIsInstanceOf` only returns **TRUE** if the class of the object is of the class *aClassObj* itself.

HOWTO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: `somobj.h`

SEE ALSO

`somDescendedFrom` -398, `somGetClass` -395, `somIsInstanceOf` -397

● somIsInstanceOf SOMObject method

Indicates if an object is an instance of a given class.

SYNTAX

int somIsInstanceOf (**SOMClass** * *aClassObj*)

PARAMETERS*aClassObj* - input

The class object to test the receiving object against.

RETURNSTRUE—The object is an instance of *aClassObj*.FALSE—The object is not an instance of *aClassObj*.**HOW TO CALL**

Call this method on any object to test its class. This method is different from `somIsA` in that it will only return TRUE if the class of the object is of the class *aClassObj* itself. `somIsA` will also return TRUE if the class of the object is a descendant of *aClassObj*.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFOInclude files: `somobj.h`**SEE ALSO**`somDescendedFrom` -398, `somGetClass` -395, `somIsA` -397

SOMClass Methods

somDescendedFrom indicates whether a class is descended from a given class (pg. 398).

● **somDescendedFrom** **SOMClass method**

Indicates whether a class is descended from a given class.

SYNTAX`int somDescendedFrom (SOMClass * aClassObj)`**PARAMETERS***aClassObj* - input

The class object to test the class object against.

RETURNS

TRUE—The class object is descended from *aClassObj*.

FALSE—Otherwise.

HOW TO CALL

Call this method on any class object to test its ancestry.

HOW TO OVERRIDE

This method is generally not overridden.

OTHER INFO

Include files: somcls.h

SEE ALSO

somGetClass -395, somIsA -397, somIsInstanceOf -397

SOMClassMgr Methods

somLocateClassFile returns the name of the module in which a class is defined (pg. 399).

● **somLocateClassFile** **SOMClassMgr method**

Returns the name of the module in which a class is defined.

SYNTAX

zString somLocateClassFile (**somId** *classId*, **long** *majorVersion*, **long** *minorVersion*)

PARAMETERS

classId - input

The SOM ID for the class. This can be obtained by calling somIdFromString.

majorVersion - input

The major version of the class.

minorVersion - input

The minor version of the class.

RETURNS

The name of the module that contains the specified class.

HOW TO CALL

Call this method on SOMClassMgrObject to obtain the module name for a class. Workplace's class manager, an instance of WPClassManager, responds by returning the module name specified when the class was registered with WinRegisterObjectClass.

HOW TO OVERRIDE

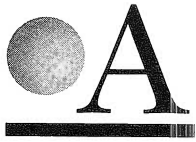
This method should not be overridden.

OTHER INFO

Include files: somcm.h

SEE ALSO

somGetClassName -396, WinRegisterObjectClass -18



Setup String Keynames

The following is a table of setup string keynames and their possible values. Setup strings are used in many Workplace methods to set object data.

Class	Keyname	Value	Description
WObject	CCVIEW	"DEFAULT"	Use the system default.
		"YES"	Allow multiple instances of views to be opened.
		"NO"	Disallow multiple instances of views.
	DEFAULTVIEW	"SETTINGS"	Use settings for the default view.
		"DEFAULT"	Use the class default view.
		n	The ID of an application-defined view.

Class	Keyname	Value	Description
WPObject	HELPLIBRARY	file name	Name of the library file to use for the general help panel. Does not need to be fully qualified if the file is in the help path.
	HELPPANEL	resource ID	The ID of the general help panel.
	HIDEBUTTON	"YES"*	Use a hide button for the views.
		"NO"	Use a minimize button.
	ICONFILE	file name	Fully qualified path of the icon.
	ICONPOS	x,y	The initial icon position in a nongridded icon view.
	ICONRESOURCE	id,module	The resource ID and module name for the icon. The .DLL must already be loaded by the Workplace process.
	MINWIN	"DESKTOP"	Minimize views to the desktop.
		"HIDE"	Hide windows when minimize button is pressed.
		"VIEWER"*	Minimize windows to the Minimized Window Viewer.
	NOCOPY	"YES"	Object cannot be copied.
		"NO"*	Object can be copied.
	NODELETE	"YES"	Object cannot be deleted.
		"NO"*	Object can be deleted.
	NODRAG	"YES"	Object's icon cannot be dragged.
		"NO"*	Object's icon can be dragged.
	NODROP	"YES"	No objects can be dropped on this object.
		"NO"*	Objects can be dropped on this object.
	NOLINK	"YES"	A shadow of this object cannot be made.
		"NO"*	A shadow of this object can be made.
	NOMOVE	"YES"	Object cannot be moved.
		"NO"*	Object can be moved.
	NOPRINT	"YES"	Object cannot be printed.
		"NO"*	Object can be printed.

*This is the default for the class

Class	Keyname	Value	Description
WPObject	NORENAME	"YES"	Object cannot be renamed.
		"NO"*	Object can be renamed.
	NOSETTINGS	"YES"	Object does not support the settings view.
		"NO"*	Object supports the settings view.
	NOTVISIBLE	"YES"	Object is invisible.
		"NO"*	Object is visible.
	OBJECTID	objectid	The persistent object ID for this object. The string must begin with '<' and end with '>' and must be unique in the system.
	OPEN	"SETTINGS"	Open the settings notebook immediately.
		"DEFAULT"	Open the default view immediately.
	TEMPLATE	"YES"	Object is a template.
		"NO"*	Object is not a template.
WPFolder	TITLE	title	The title for the object.
	ALWAYSSORT	"YES"	Always sort the views.
		"NO"*	Do not sort the views.
	BACKGROUND	f,m,s,t,c	f—fully qualified path of image file. (specify '?' for the boot drive)
			m—image mode: N for normal,
			T for tiled, S for scaled
			s—scaling factor
			t—background type: I for image,
			C for color
			c—background color: R G B or "DEFAULT."
		Example: "BACKGROUND=c:\file.bmp,N,I,C,0 0 128"	
	DEFAULTVIEW	"TREE"	Use Tree for default view.
		"ICON"	Use Icon for default view.
		"DETAILS"	Use Details for default view.
	DETAILSCLASS	classname	Class that defines details columns.

*This is the default for the class

Class	Keyname	Value	Description
WPFolder	DETAILSFONT	font	Font and point size to use for details view. Example: 8.Helv
	DETAILSVIEW	"NORMAL"	Use normal-size icons.
	DETAILSVIEW	"MINI"	Use mini icons.
	ICONFONT	font	Font and point size to use for icon view. Example: 8.Helv
	ICONVIEW	"FLOWED"	Flow the list items.
		"NONFLOWED"	Do not flow the list items.
		"NONGRID"	Icon view is nongridded.
		"NORMAL"	Use normal-size icons.
		"MINI"	Use mini icons.
		"INVISIBLE"	Hide the icons in flowed or nonflowed views.
	ICONNFILE	index,filename	index—this must be set to 1.
			filename—fully qualified path of the icon.
	ICONNRESOURCE	index,id,module	index—this must be set to 1.
			id—the ID of the icon resource.
			module—the name of the .DLL containing the resource.
	ICONVIEWPOS	x,y,cx,cy	The position and size of the Icon view.
	OPEN	"DETAILS"	Open the Details view immediately.
		"ICON"	Open the Icon view immediately.
		"TREE"	Open the Tree view immediately.
	REMOVEFONTS	"YES"	Remove all font settings from the folder.
	SORTCLASS	classname	Class that defines sort attributes.
	TREEFONT	font	Font and point size to use for Tree view. Example: 8.Helv
	TREEVIEW	"NORMAL"	Use normal-size icons.
		"MINI"	Use mini icons.
		"INVISIBLE"	Hide the icons.
		"LINES"	Use lines in Tree view.
		"NOLINES"	Do not display lines.

*This is the default for the class

Class	Keyname	Value	Description
WPPProgram and WPPProgram File	WORKAREA	"YES"	This folder is a workarea.
		"NO"*	This folder is not a workarea.
	ASSOCFILTER	filterlist	A list of extensions of data files to associate with this program. Example: *.C,*.DOC
	ASSOCTYPE	typelist	A list of types of data files to associate with this program. Example: Plain Text,C Code
	EXENAME (WPPProgram only)	filename	File name of the executable. Must be fully qualified if the program does not reside in the system PATH.
	HIDEBUTTON	not supported	
	MAXIMIZED	"YES"	Maximize the application when started.
		"NO"*	Do not maximize the application.
	MINIMIZED	"YES"	Minimize the application when started.
		"NO"*	Do not minimize the application.
	NOAUTOCLOSE	"YES"	Do not close the window when the program terminates. This is for OS/2 or DOS-windowed applications.
		"NO"*	Close the window after terminating.
	PARAMETERS	param1 param2...	Parameters to pass to the program. Each one is separated by a space.
	PROGTYPE		Program type is:
		"FULLSCREEN"	OS/2 fullscreen
		"PM"	Presentation Manager
		"PROG_31_STD"	WIN-OS/2 standard fullscreen.
		"PROG_31_STD SEAMLESS COMMON"	WIN-OS/2 standard seamless common session
		"PROG_31_STD SEAMLESSVDM"	WIN-OS/2 standard seamless separate session
		"PROG_31_ENH"	WIN-OS/2 enhanced fullscreen
		"PROG_31_ENH SEAMLESS COMMON"	WIN-OS/2 enhanced seamless common session

*This is the default for the class

Class	Keyname	Value	Description
WPPrinter		"PROG_31_ENH SEAMLESSVDM"	WIN-OS/2 enhanced seamless separate session.
		"VDM"	DOS fullscreen
		"WINDOW ABLEVIO"	OS/2 window
		"WINDOWEDVDM"	DOS window
		SET	dos setting=value The DOS setting and value for a DOS or WIN-OS/2 application. See Appendix B for a list of DOS settings. Example: "SET DOS_AUTOEXEC=C:\TEMPEXEC.BAT"
	STARTUPDIR	pathname	The working directory for this program.
	APPDEFAULT	"YES"	This is the default printer.
		"NO"	This is not the default printer.
	DEFAULTVIEW	"DETAILS"	Use Details for the default view.
		"ICON"	Use Icon for the default view.
	JOBDialog BEFOREPRINT	YES	Display the job properties dialog before printing.
		"NO"	Do not display the job properties dialog.
	JOBPROPERTIES	filename	The full path to a file containing job properties for the print object. This can be created by saving PRQINFO3->pDriverData to a file.
WPPrinter	OUTPUTTOFILE	"YES"	Output printing to a file.
		"NO"	Do not output to a file.
	PORTNAME	port(s)	The name of the port for this printer. Multiple ports are separated by commas.
	PRINTDRIVER	driver.device	The full name of an installed printer driver. Multiple drivers are separated by commas.
	PRINTER SPECIFICFORMAT	"YES"	Jobs are spooled in raw format.
		"NO"	Jobs are spooled in standard format.

*This is the default for the class

Class	Keyname	Value	Description
WPPrinter	PRINTWHILE SPOOLING	"YES"	Printing is enabled while a job is spooling.
		"NO"	Printing is not enabled while spooling.
	QSTARTTIME	time	The time, in the format HH:MM, when the printer starts printing.
	QSTOPTIME	time	The time, in the format HH:MM, when the printer stops printing.
	QUEUENAME	name	The local name for the queue. This keyname is only used when a printer is created.
	QUEUEDRIVER	driver	The name of the queue driver.
	SEPARATORFILE	filename	The full path of the separator file for this printer
WPLaunchPad	FPOBJECTS	Object,Object ...	The list of objects to insert into the Launch Pad. Object can be an object ID or the full path of a file system object.
WPDisk	DRIVENUM	n	The logical drive number (A=1,B=2, etc.).
WPSshadow	SHADOWID	Object	The object this shadow refers to. This can be an object ID or the full path of a file system object.
WPPalette	XCELLCOUNT	n	The number of columns.
	YCELLCOUNT	n	The number of rows.
	XCELLWIDTH	n	The width of the cells in pixels.
	YCELLHEIGHT	n	The height of the cells in pixels.
	XCELLGAP	n	The gap between the columns in pixels.
	YCELLGAP	n	The gap between the rows in pixels.
WPColor Palette	COLORS	color, color ...	Colors for each cell in the palette separated by commas. Each color is a hex number in the format of 0xRRGGBB (red, green, and blue values)

*This is the default for the class

Class	Keyname	Value	Description
WPFont Palette	FONTS	font, font ...	Fonts for each cell in the palette separated by commas. Each font string contains the point size and face name. Example: 10.Helv,12.Helv



DOS Settings Keynames

The following is a table of DOS settings keynames and their possible values. These settings can be passed in a setup string for a program object. For example, to set the DPMI memory limit for a program object, specify the following as part of the setup string:

```
SET DPMI_MEMORY_LIMIT=64
```

Setting	Value	Description
AUDIO_ADAPTER_SHARING	"Required"	Session-required access to audio adapter.
	"Optional"	Use audio adapter if it is available.
	"None"	No audio adapter is needed.
COM_DIRECT_ACCESS	"1"	Give session direct access to COM ports.
	"0"	Do not give session direct access to COM ports.
COM_HOLD	"1"	Keep COM ports open until session ends.
	"0"	Do not keep COM ports open.

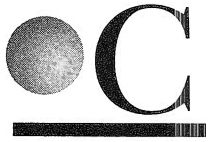
Setting	Value	Description
COM_RECEIVE_BUFFER_FLUSH	"RECEIVE DATA INTERRUPT ENABLE"	Flush the received data buffer when the program enables the receive data interrupt.
	"NONE"	Keep data in the received data buffer.
	"SWITCH TO FOREGROUND"	Flush the received data buffer when the session is switched to the foreground.
	"ALL"	Flush received data buffer in both situations.
COM_SELECT	"ALL"	Do not limit the program's access to COM ports.
	"COMx"	Limit the program's access to specified COM port.
	"NONE"	Do not allow access to COM ports.
DOS_AUTOEXEC	pathname	The fully qualified path of the batch file to execute when the session starts. C:\AUTOEXEC.BAT is the default.
DOS_BACKGROUND_EXECUTION	"1"	Allow the program to run in the background.
	"0"	Stop the program when it is not in the foreground.
DOS_BREAK	"1"	Activate Ctrl+C as a break character.
	"0"	Do not activate Ctrl+C as a break character.
DOS_DEVICE	pathname, pathname,...	The fully qualified path for each DOS device driver to load for this session. Separate each one with a comma.
DOS_FCBS	num	The maximum number of file control blocks that can be opened in this session. This affects file-sharing modules (range 0-255).
DOS_FCBS_KEEP	num	The number of file control blocks to set aside so they cannot be closed by the operating system (range 0-255).
DOS_FILES	num	The maximum number of file handles that can be opened in this session.

Setting	Value	Description
DOS_HIGH	"1"	Load the DOS kernel above the 1MB memory address.
	"0"	Do not load the DOS kernel above the 1MB memory address.
DOS_LASTDRIVE	letter	The highest drive available to this session. Default is Z.
DOS_RMSIZE	num	The amount of memory available to this session. Range is 128K–640K; default is 640.
DOS_SHELL	filename	The path and file name for the command processor for this session.
DOS_STARTUP_DRIVE	letter or filename	The drive letter or an image created with VMDISK to boot a specific version of DOS.
DOS_UMB	"1"	Allow DOS TSRs and devices to be loaded with LOADHIGH and DEVICEHIGH.
	"0"	Give ownership of upper memory blocks to TSRs and devices
DOS_VERSION	version,version, version...	Each version string is specified as: ProgramName^,Version^,SubVersion^, repeat where repeat is the number of times this version can be reported to the program (255 for indefinite). For example: "SET DOS_VERSION= PROG1.EXE^,3^,30^,255, PROG2.EXE^,3^,30^,255" to report to Prog1.exe and Prog2.exe that the version of DOS is 3.3.
DPMI_DOS_API	"AUTO"	The program comes with an extender based on DPMI and will perform memory translations.
	"ENABLED"	The operating system will perform the translation.
	"DISABLED"	The program does not use DPMI.

Setting	Value	Description
DPMI_MEMORY_LIMIT	num	The amount of DPMI available to this session (range 0–512MB).
DPMI_NETWORK_BUFF_SIZE	num	The size of the network translation buffer for this DPMI program (range 1–64KB).
EMS_FRAME_LOCATION	“AUTO” “NONE” location	The session will locate expanded memory automatically. Turn off the high EMS range. The location of the 64KB EMS frame. For example: “SET EMS_FRAME_LOCATION=C400”
EMS_HIGH_OS_MAP_REGION	num	The amount of memory this program can add to the EMS (range 0–96KB).
EMS_LOW_OS_MAP_REGION	num	The size of the remappable conventional memory available to this session (range 0–576KB).
EMS_MEMORY_LIMIT	num	The amount of EMS memory available to this session (range 0–32768KB).
HW_NOSOUND	“1” “0”	Prevent programs in this session from generating sounds. Allow programs to generate sounds.
HW_ROM_TO_RAM	“1” “0”	Copy ROM BIOS code to RAM. Otherwise
HW_TIMER	“1” “0”	Give programs in this session direct access to hardware timer ports. Otherwise
IDLE_SECONDS	num	A period of allowable idle time before the operating system reduces processor time for this program (range 0–60 seconds).
IDLE_SENSITIVITY	percentage	Threshold for polling time before the operating system reduces processor time for this program (set to 100 to disable idle detection).

Setting	Value	Description
INT_DURING_TO	"1"	Allow interrupts during file reads and writes.
	"0"	Otherwise
KBD_ALTHOME_BYPASS	"1"	Prevent switching from fullscreen to windowed when Alt+Home is pressed.
	"0"	Otherwise
KBD_BUFFER_EXTEND	"1"	Increase the size of the keyboard typeahead buffer.
	"0"	Do not extend the keyboard typeahead buffer.
KBD_CTRL_BYPASS	"NONE"	Do not bypass any control-key sequences
	"ALT_ESC"	Allow the program to process Alt+Esc.
	"CTRL_ESC"	Allow the program to process Ctrl+Esc.
KBD_RATE_LOCK	"1"	Prevent the program from changing the system keyboard repeat rate.
	"0"	Otherwise
MEM_EXCLUDE_REGIONS	region^, region^,...	Regions EMS/XMS cannot use because they are needed for device drivers.
MEM_INCLUDE_REGIONS	region^, region^,...	Regions EMS/XMS can use between RMSIZE and IMB.
MOUSE_EXCLUSIVE_ACCESS	"1"	Give this session exclusive ownership of the mouse.
	"0"	Otherwise
PRINT_SEPARATE_OUTPUT	"1"	Send separate printer output if two programs in this session send to the same device.
	"0"	Do not send separate output in this case.
PRINT_TIMEOUT	num	The timeout before the operating system forces a print job to the printer (range 0–3600 seconds).
SESSION_PRIORITY	num	The priority level for this session. Range is 1–32 (lowest to highest).

Setting	Value	Description
VIDEO_8514A_XGA_IOTRAP	"1"	Allow controlled access to the video device.
	"0"	Provide unrestricted access.
VIDEO_FASTPASTE	"1"	Speed up input from sources other than the keyboard.
	"0"	Otherwise
VIDEO_MODE_RESTRICTION	"NONE"	Do not limit video mode support
	"CGA"	Limit video mode support for CGA graphics.
	"MONO"	Limit video mode support for text.
VIDEO_ONDEMAND_MEMORY	"1"	Delay allocation of video-save buffers.
	"0"	Otherwise
VIDEO_RETRACE_EMULATION	"1"	Emulate text-based video ROM functions.
	"0"	Otherwise
VIDEO_SWITCH_NOTIFICATION	"1"	Notify the program when the session switches in and out of fullscreen mode.
	"0"	Otherwise
VIDEO_WINDOW_REFRESH	num	The window update frequency. Range is 1–600 tenths of a second.
XMS_HANDLES	num	The number of handles needed to identify XMS blocks (range 0–128).
XMS_MEMORY_LIMIT	num	The amount of available XMS memory (range 0–16384 KB).
XMS_MINIMUM_HMA	num	The minimum allocation size of the high memory area (range 0–63KB).



Settings Page Methods

The predefined Workplace Shell classes provide instance methods for adding each page to their settings notebooks. These methods can be called from the `wpAddSettingsPages` (pg. 181) override or can be removed by overriding the individual page method and returning `SETTINGS_PAGE_REMOVED`. Each settings page instance method has the same parameters and return code. For example, the definition of `wpAddObjectGeneralPage` is:

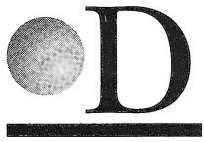
```
ULONG wpAddObjectGeneralPage(HWND hwndNotebook )
```

where *hwndNotebook* is the notebook window handle passed to the `wpAddSettingsPages` override and the return code is the page ID returned from `wpInsertSettingsPage` (pg. 183). The following is a list of the settings page methods for each class:

Class	Method
WPObject	<code>wpAddObjectGeneralPage</code> <code>wpAddObjectGeneralPage2</code> <code>wpAddObjectWindowPage</code>
WPFileSystem	<code>wpAddFile1Page</code>

Class	Method
WPFileSystem	wpAddFile2Page wpAddFile3Page wpAddFileMenuPage
WPDataFile	wpAddFileTypePage
WPFolder	wpAddFolderBackgroundPage wpAddFolderIncludePage wpAddFolderSelfClosePage wpAddFolderSortPage wpAddFolderView1Page wpAddFolderView2Page wpAddFolderView3Page
WPDesktop	wpAddDesktopLockup1Page wpAddDesktopLockup2Page wpAddDesktopLockup3Page wpAddDesktopArcRest1Page wpAddDesktopDefDT1Page
wpProgram and wpProgramFile	wpAddProgramAssociationPage wpAddProgramPage wpAddProgramSessionPage
WPLaunchPad	wpAddLaunchPadPage1 wpAddLaunchPadPage2
WPClock	wpAddClockAlarmPage wpAddClockDateTimePage wpAddClockView1Page wpAddClockView2Page
WPCountry	wpAddCountry wpAddCountryNumbersPage wpAddCountryPage wpAddCountryTimePage
WPDisk	wpAddDiskDetailsPage
WPKeyboard	wpAddKeyboardMappingsPage wpAddKeyboardSpecialNeedsPage wpAddKeyboardTimingPage
WPMouse	wpAddMouseCometPage wpAddMouseMappingsPage wpAddMousePtrPage

Class	Method
WPMouse	wpAddMouseTimingPage
	wpAddMouseTypePage
WPSound	wpAddSoundWarningBeepPage
WPSystem	wpAddSysFdrDefViewPage
	wpAddSysFdrSelfClosePage
	wpAddSystemConfirmationPage
	wpAddSystemInputPage
	wpAddSystemLogoPage
	wpAddSystemPrintScreenPage
	wpAddSystemScreenPage
	wpAddSystemWindowPage
	wpAddTitleConfirmationPage

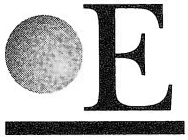


Workplace Default Object IDs

The following is a list of object IDs for objects created by the system. These default Workplace objects always have object IDs that begin with WP_.

Object ID	Description
<WP_DESKTOP>	Desktop folder
<WP_NOWHERE>	Nowhere folder: an invisible folder for placing objects that should not be in a folder.
<WP_TEMPS>	Templates folder
<WP_VIEWER>	Minimized Window Viewer
<WP_SHRED>	Shredder
<WP_MINDEX>	Master Help Index
<WP_INFO>	Information Folder
<WP_NETWORK>	Network Folder
<WP_GLOSS>	Glossary
<WP_TUTOR>	Tutorial

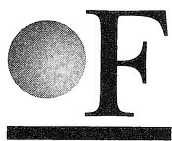
Object ID	Description
<WP_CMDREF>	Command Reference
<WP_OS2SYS>	OS/2 System folder
<WP_DRIVES>	Drives folder
<WP_START>	Startup folder
<WP_CONFIG>	System Setup folder
<WP_CLOCK>	Clock
<WP_KEYB>	Keyboard
<WP_MOUSE>	Mouse
<WP_SOUND>	Sound
<WP_SYSTEM>	System
<WP_CNTRY>	Country
<WP_FNTPAL>	Font Palette
<WP_CLRPAL	Color Palette
<WP_SCHPAL>	Scheme Palette
<WP_PROMPTS>	Command Prompts
<WP_GAMES>	Games folder
<WP_TOOLS>	Productivity
<WP_LAUNCHPAD>	Launch Pad



Predefined Views

The following is a list of views supported by the system.

Class	View	Value
WPObject	OPEN_SETTINGS	2
WPFolder	OPEN_CONTENTS	1
	OPEN_TREE	101
	OPEN_DETAILS	102
WPDisk	OPEN_AUTO	120
WPProgram and WPProgramFile	OPEN_RUNNING	4
WPPalette	OPEN_PALETTE	121



Device Object Settings

The following list contains the device object settings for WPSystem, WPMouse, and WPKeyboard. These settings can be queried and set with wpclsQuerySetting (pg. 126) and wpclsSetSetting (pg. 131).

Class	Setting	Value
WPSystem	Animation	ANIMATION_ON—Enable window animation. ANIMATION_OFF—Disable window animation. ANIMATION_DEFAULT—Use the system default.
	ConfirmCopyMove CreateShadow	CONFIRM_ON—Confirm copy, move, and create shadow operations. CONFIRM_OFF—Do not confirm these operations. CONFIRM_DEFAULT—Use the system default.

Class	Setting	Value
WPSystem	ConfirmFolderDelete	CONFIRM_ON—Confirm folder deletion. CONFIRM_OFF—Do not confirm folder deletion. CONFIRM_DEFAULT—Use the system default.
	ConfirmObjectDelete	CONFIRM_ON—Confirm object deletion. CONFIRM_OFF—Do not confirm object deletion. CONFIRM_DEFAULT—Use the system default.
	ConfirmRenameFileExtension	CONFIRM_ON—Display confirmation dialog when user changes the extension of a file. CONFIRM_OFF—Do not display the confirmation dialog. CONFIRM_DEFAULT—Use the system default.
	ConcurrentView	CCVIEW_ON—Allow concurrent views. CCVIEW_OFF—Disallow concurrent views. CCVIEW_DEFAULT—Use the system default.
	DisplayProgressIndicator	DISPLAY_ON—Display the progress indicator for copy, move, and delete operations. DISPLAY_OFF—Do not display the progress indicator. DISPLAY_DEFAULT—Use the system default.
	LogoDisplayTime	LOGO_INDEFINITE—Display logos until they are dismissed. LOGO_NONE—Do not display application logos. n—The amount of time, in milliseconds, to display logos.
	MinButtonAppearance	BUTTON_HIDE—Use hide buttons for views. BUTTON_MINIMIZE—Use minimize buttons for views. BUTTON_DEFAULT—Use the system default.

Class	Setting	Value
WPSystem	NameClash	NAMECLASH_PROMPT—Prompt the user when an object title clash occurs. NAMECLASH_RENAME—Rename an object when title clash occurs. NAMECLASH_REPLACE—Replace the duplicate object when a title clash occurs.
	PrintScreen	PRINTSCREEN_ON—Enable print screen. PRINTSCREEN_OFF—Disable print screen.
WPMouse	ButtonSetup	BUTTONS_LEFTHANDED—Left-handed mouse. BUTTONS_RIGHTHANDED—Right-handed mouse.
	DisplayWindowListButton	High word: INP_NONE—Required if default not used. Low word: WM_BUTTON2CLICK or WM_CHORD WINDOWLISTBUTTON_DEFAULT—Use the system default.
	DoubleClickTime	n—Time between mouse clicks. Range is 170–1060. DOUBLECLICK_DEFAULT—Use the system default.
	DragObjectButton	High word: INP_NONE—Required if default not used. Low word: WM_BUTTON1MOTIONSTART or WM_BUTTON2MOTIONSTART DRAGBUTTON_DEFAULT—Use the system default.
	EditTitleTextButton	High word: INP_ALT—Key to press with mouse button to edit title text. INP_CTRL INP_NONE INP_SHIF Low word: WM_BUTTON1CLICK WM_BUTTON2CLICK WM_BUTTON1DBLCLK WM_BUTTON2DBLCLK

Class	Setting	Value
		TEXTEDITBUTTON_DEFAULT—Use the system default.
	PopupMenuButton	<p>High word: INP_ALT—Key to press with mouse button to display a pop-up menu.</p> <p>INP_CTRL INP_NONE INP_SHIFT</p> <p>Low word: WM_BUTTON1CLICK WM_BUTTON2_CLICK WM_BUTTON2DBLCLK WM_CHORD</p> <p>POPUPBUTTON_DEFAULT—Use the system default.</p>
	TrackingSpeed	n—Mouse tracking speed (range 1-7).
WPKeyboard	CursorBlinkRate	n—Cursor blink rate. Range is CURSORBLINK_MIN to CURSORBLINK_MAX. CURSORBLINK_DEFAULT—Use the system default.
	EditTitleTextKey	<p>High word: (OR flags together)</p> <p>AF_VIRTUALKEY Required if default not used.</p> <p>AF_KBDCOMMAND Required if default not used.</p> <p>AF_ALT Include Alt in key sequence.</p> <p>AF_SHIFT Include Shift in key sequence.</p> <p>AF_CONTROL Include Ctrl in key sequence.</p> <p>Low word: VK_*—Key to start title text editing.</p> <p>TEXTEDITKEY_DEFAULT—Use the system default.</p>

Class	Setting	Value
WPKeyboard	KeyRepeatRate	n—Key repeat rate. Range is REPEATRATE_MIN to REPEATRATE_MAX. REPEATRATE_DEFAULT—Use the system default.
	KeyRepeatDelay	n—Key repeat delay. Range is REPEATDELAY_MIN to REPEATDELAY_MAX. REPEATDELAY_DEFAULT—Use the system default.
	PopupMenuKey	High word: (OR flags together) AF_VIRTUALKEY—Required if default not used. AF_KBDCOMMAND—Required if default not used. AF_ALT—Include Alt in key sequence. AF_SHIFT—Include Shift in key sequence. AF_CONTROL—Include Ctrl in key sequence. Low word: VK_*—Key to summon pop-up menus. POPUPKEY_DEFAULT—Use the system default.



Index

<class name>GetData, 392
<class name>MethodDebug, 393
<class name>NewClass, 394
_<class name>, 395

A

ACTION_BUTTONS_* values, 304
ACTIONS, 323
Associations, 269, 270, 271–276, 280–282
ASSOCTABLE, 269

B

Base class, 3
BKA_* flags, 188

C

CANCEL_DELETE, 41, 52
CCVIEW_ values, 164
CELL, 335
CFA_* flags, 132–133
Class(es)
 data, *see* class data
 deregistering, 12, 28
 enumerating, 14, 32
 icons, 105, 124, 130, 260, 262
 installing, 6
 locking, 70
 method, 4
 module, 373, 399
 pointer, 4
 querying, 395–398
 registering, 33
 replacing, 19, 375
 styles, 127
 title, 104, 129
 unlocking, 69
 usage count, 69, 70
Class data, 78
 initializing, 79, 101
 saving, 79
 uninitializing, 79, 102

CLASSFIELDINFO, 132
CLSSTYLE_* flags, 127–128
CMP_* values, 135
COMPARE_SUPPORTED, 134
CONFIRM_DELETE, 41, 52
CONFIRM_DELETEFOLDER, 41, 52
CONFIRM_* flags, 111
Container records, 107, 112
CO_* values, 10
CRA_* flags, 107
CTXT_* flags, 193–194
CV_* flags, 246

D

Data files, *see* WPDataFile
DEFAULTBUTTON, 162
Desktop, *see* WPDesktop
Details data:
 column visibility, 239, 247, 252, 254
 refreshing, 106
 sample code, 138–143
 sorting, 240, 250, 257, 258
 specifying, 113, 123
Device class, 4, 105, 126, 131, 421
Device object, *see* Device class
DEVOPENSTRUC, 370
Dirty objects, 79
Disk objects, *see* WPDisk
DM_DRAGOVER, *see* wpDragOver
DM_DROP, *see* wpDrop
DO_* flags, 203
DOR_* values, 203
DOS Settings, 409
Drag/Drop, 200
 from Workplace, 201
 to Workplace, 201
DRAGINFO, 211
DRAGITEM, 212
DRF_OBJECT, 201
Drive objects, *see* WPDisk
DRM_OBJECT, 201

E

Error handling, 371, 374–375, 384–385
Exit list, 103
Extended Attributes, 213

F

FDATE, 228
FILE_* flags, 216
Finding objects, 371, 376–383
Folder, *see* WPFolder
FTIME, 228

H

Handles, *see* object handles
Help, 337
 default, 340–341, 342
 displaying, 338
 for pop-up menus, 339
 for settings notebooks, *see* PAGEINFO
HIDEBUTTON, 162

I

ICONINFO, 136
Installation, 6
Instance data, 78
 accessing, 392
 allocating, 79, 80
 freeing, 79, 82
 initializing, 79, 83
 restoring, 79, 84, 87–91
 saving, 21, 33, 79, 92–100
 uninitializing, 79, 100
In-use list, 144, 157

L

Launch Pad, *see* WPLaunchPad
LINKITEM, 153

M

MEMORYITEM, 153
MENU_* values, 192
MINBUTTON, 162
MINWIN_* values, 166

N

NO_DELETE, 41, 52
Notebook, *see* Settings notebook
Nowhere folder, 5, 234, 301

O

OBJCLASS, 36
Object(s):
 asleep, *see* object, dormancy
 awake, 38, 73
 copying, 8, 24, 44
 creating, 9, 25, 47, 71
 data, *see* instance data
 deleting, 13, 29, 51, 53–54
 dormancy, 38–39
 handles, 7, 17, 60, 72
 icons, 105, 114, 115, 119, 120
 IDs, 7–8, 117, 418
 lock count, 38, 55
 locking, 38–39, 55, 56
 moving, 16, 30, 57
 opening, *see* views, opening
 printing, 344, 350
 saving, *see* instance data, saving
 settings, 104
 styles, 108, 118, 121
 title, 104, 118, 122
 unlocking, 38–39, 65
OBJECT_FROM_PREC, 234
OBJECTSNOOZETIME, 38
OBJSTYLE_* flags, 108–109
OBJSTYLE_NOSETTINGS, 109, 110
OD_* values, 369
OK_DELETE, 41, 52
OPEN_USER, 156, 160
OR_* flags, 59

P

PAGEINFO, 187
Palettes, *see* WPPalette
PALINFO, 335
PD_JOB_PROPERTY, 369
Persistent handles, *see* object handles
PMSHELL.EXE, 2, 7
PO_* flags, 367
POINTL, 137
Pop-up menus, 190
 adding items, 190, 195, 199
 displaying, 191
 filtering items, 190, 193
Presentation Manager, 1
PRINTDEST, 369
Printers, *see* WPPrinter
PROGDETAILS, 284
Program objects, *see* WPProgram(File)

PROGTYPE, 284
PROG_* values, 285

Q

QC_* values, 244

R

RC_DROP_* values, 207
RECORDINSERT, 267
RECORDITEM, 153
RECTL, 336
*_RENAMEFILESWITHTEXT values, 271
REXX, 6–8, 23

S

Settings notebook, 180
 adding pages, 180
 removing pages, 181, 415
 sizing, 185–186
Settings page methods, 415
SETTINGS_PAGE_NUMBERS, 188
SETTINGS_PAGE_REMOVED, 181, 415
Setup strings, 61, 63, 64, 401
Shadow objects, *see* WPSshadow
SHE_* flags, 285
SIZEL, 189
somDescendedFrom, 398
SOMError, 386
somFree, 387
somGetClass, 5, 395
somGetClassName, 396
somIdFromString, 389
somInit, 387
SOMInitModule, 387
somIsA, 397
somIsInstanceOf, 397
somLocateClassFile, 399
somLPrintf, 390
somNew, 387
SOMOutCharRoutine, 388
somPrintf, 391
somRegisterClass, 387
SOMSTARS, 387
somUnInit, 387
SORTBY_SUPPORTED, 134
Sorting, *see* Details data, sorting
Source rendering, *see* wpFormatDragItem
Storage class, 3
SysCopyObject, 24
SysCreateObject, 25

SysCreateShadow, 27
SysDeregisterObjectClass, 28
SysDestroyObject, 29
SysMoveObject, 30
SysOpenObject, 31
SysQueryClassList, 32
SysRegisterObjectClass, 33
SysSaveObject, 33
SysSetObjectData, 34
System Object Model, 1, 386

T

Templates, 43, 48, 67

U

USAGE_LINK, 51, 146, 148, 150, 297
USAGE_MEMORY, 81, 83, 146, 150
USAGE_OPENFILE, 146, 150
USAGE_OPENVIEW, 144, 146, 148, 150
USAGE_RECORD, 146, 148, 150
USEITEM, 144, 154
USERWORD_FROM_PREC, 234

V

VIEWFILE, 154
VIEW_* flags, 152
VIEWITEM, 155
Views, 156
 closing, 158
 concurrent, 164, 170
 default, 157, 165, 171, 178
 fonts, *see* WPFold fonts
 hiding, 159
 minimize behavior, 166, 172
 minimize button appearance, 162, 169, 177
 opening, 16, 31, 157, 160, 164, 170, 174
 predefined, 420
 registering, 157, 167
 restoring, 168, 173
VIEWSTATE_* flags, 155

W

WinCopyObject, 8
WinCreateObject, 7, 9
WinCreateShadow, 11
WinDeregisterObjectClass, 12
WinDestroyObject, 13
WinEnumObjectClasses, 14
WinMoveObject, 16
WinOpenObject, 16

- WinQueryObject, 17
- WinRegisterObjectClass, 2, 6, 11, 18
- WinReplaceObjectClass, 4, 19
- WinSaveObject, 21
- WinSetObjectData, 22
- Workplace Classes, 2–4
- WPAbstract, 3–4, 38, 79
- wpAddFirstChild, 231
- wpAddSettingsPages, 180, 181
- wpAddToContent, 232
- wpAddToObjUseList, 144, 145, 156
- wpAllocMem, 80
- WPClock, 3–4
- wpClose, 158
- wpclsCreateDefaultTemplates, 67
- wpclsDecUsage, 69
- wpclsFindObjectEnd, 376
- wpclsFindObjectFirst, 377
- wpclsFindObjectNext, 382
- wpclsFindObjectOne, 383
- wpclsInitData, 101
- wpclsInsertMultipleObjects, 264
- wpclsNew, 71
- wpclsObjectFromHandle, 72
- wpclsQueryActiveDesktop, 259
- wpclsQueryActiveDesktopHWND, 260
- wpclsQueryAwakeObject, 73
- wpclsQueryButtonAppearance, 177
- wpclsQueryDefaultHelp, 342
- wpclsQueryDefaultView, 178
- wpclsQueryDetailsInfo, 123
- wpclsQueryEditString, 334
- wpclsQueryError, 384
- wpclsQueryFolder, 74
- wpclsQueryIcon, 105
- wpclsQueryIconData, 124
- wpclsQueryIconDataN, 260
- wpclsQueryIconN, 262
- wpclsQueryInstanceFilter, 227
- wpclsQueryInstanceType, 226
- wpclsQueryObject, 75
- wpclsQueryObjectFromFrame, 179
- wpclsQueryObjectFromPath, 76
- wpclsQueryOpenFolders, 262
- wpclsQuerySetting, 4, 126, 421
- wpclsQuerySettingsPageSize, 185
- wpclsQueryStyle, 127
- wpclsQueryTitle, 129
- wpclsRemoveObjects, 265
- wpclsSetError, 385
- wpclsSetIcon, 130
- wpclsSetIconData, 130
- wpclsSetSetting, 4, 131, 421
- wpclsSetSettingsPageSize, 186
- wpclsUnInitData, 102
- wpCnrInsertObject, 233
- wpCnrRefreshDetails, 106
- wpCnrRemoveObject, 234
- wpCnrSetEmphasis, 107
- wpConfirmDelete, 40
- wpConfirmKeepAssoc, 270
- wpContainsFolders, 235
- wpCopiedFromTemplate, 43
- wpCopyObject, 44
- WPCountry, 3–4
- wpCreateAnother, 47
- wpCreateFromTemplate, 48
- wpCreateShadowObject, 50
- wpCreateShadowObjectExt, 296
- WPDataFile, 214
 - associations, *see* Associations
 - icon, 271, 280
 - printing, 349–362
- wpDelete, 51
- wpDeleteAllJobs, 346
- wpDeleteContents, 236
- wpDeleteFromContent, 237
- wpDeleteFromObjUseList, 148
- WPDesktop, 238, 259, 260
- WPDisk, 286
 - ejecting, 287
 - icon, 293
 - locking, 289, 291
 - logical drive, 292
 - querying, 290
 - root folder, *see* WPRootFolder
- wpDisplayHelp, 338
- wpDisplayMenu, 191
- wpDragCell, 327
- wpDraggedOverObject, 202
- wpDragOver, 204
- wpDrop, 206
- wpDroppedOnObject, 209
- wpEditCell, 328
- wpEndConversation, 202
- WPFileSystem, 3–4, 38, 79, 213
- wpFilterPopupMenu, 193
- wpFindMinWindow, 372
- wpFindUseItem, 150
- wpFindViewItem, 152

- WPFolder, 213, 229
 - animation icon, 260, 262
 - contents, 229, 232, 235–237, 242–243
 - details class, 247, 254
 - folder flags, 241, 248, 255
 - fonts, 248, 256
 - icon positions, 251
 - querying, 251
 - refreshing, 222
 - sort class, 250, 257
 - view attributes, 245, 253
- wpFormatDragItem, 209
- wpFree, 53
- wpFreeMem, 82
- wpHide, 159
- wpHoldPrinter, 346
- wpInitData, 83
- wpInsertPopupMenuItems, 195
- wpInsertSettingsPage, 180, 183
- wpIsCurrentDesktop, 238
- wpIsDeleteable, 54
- wpIsDetailsColumnVisible, 239
- wpIsDiskSwapped, 288
- wpIsLocked, 55
- wpIsObjectInitialized, 56
- wpIsSortAttribAvailable, 240
- wpJobAdded, 347
- wpJobChanged, 348
- wpJobDeleted, 348
- WPKeyboard, 3–4
- WPLaunchPad, 301
 - action buttons, 303–304, 313
 - closing drawers, 305, 314
 - contents, 311, 320–322
 - frame controls, 310, 319
 - icon size, 304–305, 314
 - orientation, 308–309, 317–318
 - refreshing, 312
 - text, 306–307, 315–316
- wpLockDrive, 289
- wpLockObject, 39, 56
- WPMENUID_* values, 196
- wpMenuItemHelpSelected, 339
- wpMenuItemSelected, 198
- wpModifyFldrFlags, 241
- wpModifyPopupMenu, 199
- wpModifyStyle, 108
- wpModuleForClass, 373
- WPMouse, 3–4
- wpMoveObject, 57
- WPObject, 2
- wpObjectReady, 58
- wpOpen, 160
- wpPaintCell, 329
- WPPalette, 325
- wpPopulate, 242
- WPPower, 3–4
- WPPrinter, 344
 - computer name, 363
 - creating, 345
 - default, 366
 - jobs, 346–348
 - physical name, 363
 - queue, 346, 362, 364, 367
- wpPrintMetaFile, 349
- wpPrintObject, 350
- wpPrintPiffFile, 358
- wpPrintPlainTextFile, 359
- wpPrintPrinterSpecificFile, 360
- wpPrintUnknownFile, 361
- WPProgram(File), 268
 - associations, *see* Associations
 - details, 277, 283
 - screen group ID, 279
- wpQueryActionButtons, 303
- wpQueryActionButtonStyle, 304
- wpQueryAssociatedFileIcon, 271
- wpQueryAssociatedProgram, 269, 272
- wpQueryAssociationFilter, 275
- wpQueryAssociationType, 276
- wpQueryAttr, 215
- wpQueryButtonAppearance, 162
- wpQueryCloseDrawer, 305
- wpQueryComputerName, 363
- wpQueryConcurrentView, 164
- wpQueryConfirmations, 111
- wpQueryContainer, *see* wpQueryFolder
- wpQueryContent, 243
- wpQueryCoreRecord, 112
- wpQueryCreation, 216
- wpQueryDefaultHelp, 340
- wpQueryDefaultView, 165
- wpQueryDisk, 290
- wpQueryDisplaySmallIcons, 305
- wpQueryDisplayText, 306
- wpQueryDisplayTextInDrawers, 307
- wpQueryDisplayVertical, 308
- wpQueryDrawerHWND, 308
- wpQueryDriveLockStatus, 291
- wpQueryEASize, 217

- wpQueryError, 374
- wpQueryFileName, 214
- wpQueryFileSize, 218
- wpQueryFldrAttr, 245
- wpQueryFldrDetailsClass, 247
- wpQueryFldrFlags, 248
- wpQueryFldrFont, 248
- wpQueryFldrSortClass, 250
- wpQueryFloatOnTop, 309
- wpQueryFolder, 251
- wpQueryHandle, 60
- wpQueryHideLaunchPadFrameCtrls, 310
- wpQueryIcon, 114
- wpQueryIconData, 115
- wpQueryLastAccess, 219
- wpQueryLastWrite, 219
- wpQueryLogicalDrive, 292
- wpQueryMinWindow, 166
- wpQueryNextIconPos, 251
- wpQueryObjectID, 117
- wpQueryObjectList, 311
- wpQueryPaletteInfo, 330
- wpQueryPrinterName, 363
- wpQueryProgDetails, 277
- wpQueryQueueOptions, 364
- wpQueryRealName, 220
- wpQueryRemoteOptions, 365
- wpQueryRootFolder, 293
- wpQueryScreenGroupID, 279
- wpQueryShadowedObject, 298
- wpQueryStyle, 118
- wpQueryTitle, 118
- wpQueryType, 221
- wpRedrawCell, 330
- wpRefresh, 222
- wpRefreshDrawer, 312
- wpRegisterView, 156, 167
- wpReleasePrinter, 362
- wpRender, 202
- wpRenderComplete, 202
- wpReplacementIsInEffect, 375
- wpRestore, 168
- wpRestoreData, 84
- wpRestoreLong, 87
- wpRestoreState, 89
- wpRestoreString, 90
- WPRootFolder, 286
 - querying from WPDisk, 293
- wpSaveData, 92
- wpSaveDeferred, 79, 94
- wpSaveImmediate, 79, 95
- wpSaveLong, 96
- wpSaveState, 97
- wpSaveString, 98
- wpScanSetupString, 61
- wpSelectCell, 331
- wpSetActionButtonStyle, 313
- wpSetAssociatedFileIcon, 280
- wpSetAssociationFilter, 281
- wpSetAssociationType, 282
- wpSetAttr, 223
- wpSetButtonAppearance, 169
- wpSetCloseDrawer, 314
- wpSetConcurrentView, 170
- wpSetCorrectDiskIcon, 293
- wpSetDefaultHelp, 341
- wpSetDefaultPrinter, 366
- wpSetDefaultView, 171
- wpSetDetailsColumnVisibility, 252
- wpSetDisplaySmallIcons, 314
- wpSetDisplayText, 315
- wpSetDisplayTextInDrawers, 316
- wpSetDisplayVertical, 317
- wpSetDrawerHWND, 317
- wpSetError, 375
- wpSetFldrAttr, 253
- wpSetFldrDetailsClass, 254
- wpSetFldrFlags, 255
- wpSetFldrFont, 256
- wpSetFldrSortClass, 257
- wpSetFloatOnTop, 318
- wpSetHideLaunchPadFrameCtrls, 319
- wpSetIcon, 119
- wpSetIconData, 120
- wpSetLinkToObject, 299
- wpSetMinWindow, 172
- wpSetNextIconPos, 231
- wpSetObjectID, 118
- wpSetObjectListFromHObjects, 320
- wpSetObjectListFromObjects, 321
- wpSetObjectListFromStrings, 322
- wpSetPaletteInfo, 332
- wpSetProgDetails, 283
- wpSetQueueOptions, 367
- wpSetRealName, 215
- wpSetRemoteOptions, 368
- wpSetShadowTitle, 300
- wpSetSortAttribAvailable, 258
- wpSetStyle, 121
- wpSetTitle, 122

- wpSetType, 224
- wpSetup, 63
- wpSetupCell, 333
- wpSetupOnce, 64
- WPSshadow, 295
 - creating, 11, 27, 50, 296
 - original object, 295, 298
 - setting a link, 299
 - title, 300
- WPSHredder, 3–4
- WPSound, 3–4
- wpSwitchTo, 157, 173
- WPSsystem, 3–4, 164, 170, 172
- WPTransient, 3–4, 39, 46
- wpUnInitData, 100
- wpUnlockObject, 39, 65
- wpVerifyUpdateAccess, 225
- wpViewObject, 174
- wpWaitForClose, 176

541028



The programmer's dream— OS/2® Workplace Shell methods organized for the way you work

Are you fed up with searching back and forth through clumsy manuals every time you want to look up a sequence of OS/2 functions? Relax, your life is about to get a lot easier. This practical, easy-to-use reference provides quick access to all the OS/2 Workplace Shell information you need on a daily basis. Functions are organized to help you create and customize applications with concise and complete information, and graphic features such as targeted bolding and bulleting draw your attention to the most important information first. So don't waste valuable programming time with a book in your lap—use the reference that brings you:

- Information arranged by topic
- Concise reviews of each topic or concept that explain methods and how they fit together
- Special graphic features that emphasize the most important information
- In-depth information not documented in any other book

This book and its companion volumes, *OS/2 Warp Control Program API* and *OS/2 Warp Presentation Manager API*, offer such fast and easy access to every major OS/2 API, you'll wonder how you ever got along without them.

MINDY POLLACK was an original member of the IBM Workplace Shell development team. She is now a developer with Wall Data, Inc.

MARC STOCK is a programmer/analyst with Keane, Inc. He specializes in designing and writing OS/2 applications.

Cover Design/Illustration: Nostradamus Advertising, Inc.

John Wiley & Sons, Inc.
Professional, Reference and Trade Group
605 Third Avenue, New York, N.Y. 10158-0012
New York • Chichester • Brisbane • Toronto • Singapore

ISBN 0-471-03872-5

